International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-2, Issue-4)

Research Article | **April 2013**

A survey of Data Mining Tools for Platform Dependency

Mrs.Priti Lale(Lahane)
M.E (CSE-Persuing)
PRMIT&R,Badnera, Amaravati.
India.

Mr.Shrikant Akarte
Asst.Prof. in Comp.Dept.
PRMIT&R,Badnera, Amaravati.
India.

Abstract:

Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner. However, many data mining tools are available on the market each having their pros and cons in terms of reliability, usability, security, and performance. This paper presents a comparative study on the recent data mining tools for platform dependency. They are mainly SEE5 and C5.0. The testing is done by implementing a sample application by using these tools. This would pave the way to build a head-to-head comparative evaluation that shows the important feature that is comprehensibility of these tools which make them different from other data mining tools.

Keywords:- SEE5,C5.0,Classifier,Rulsets,Decision Tree.

I. Introduction

Data mining is all about extracting patterns from an organization's stored or warehoused data. These patterns can be used to gain insight into aspects of the organization's operations, and to predict outcomes for future situations as an aid to decision making. See5(Windows Xp/Vista/7/8) and its Unix counterpart C5.0 are sophisticated data mining tools for discovering patterns that delineate categories, assembling them into classifiers, and using them to make predictions. Patterns often concern the categories to which situations belong. See5/C5.0 has been designed to analyze substantial databases containing thousands to millions of records and tens to hundreds of numeric, time, date, or nominal fields. See5/C5.0 also takes advantage of computers with up to eight cores in one or more CPUs (including Intel Hyper-Threading) to speed up the analysis.

II. Background

This section discusses the history, versions, and features of the data mining tools under test. They are respectively SEE5 and C5.0.

2.1 SEE5

See5 from RuleQuest is a linear classifier with a large range of practical uses. See5 is very useful if used in the situations for which it is intended. There are many data mining tools, classification applications, and applications which could be said to use artificial intelligence. See5 from Rule Quest, which is based on an extension of the C4.5 algorithm, is one such application. See5 finds patterns in data bases and can be used for a variety of tasks including data mining, machine learning, pattern recognition, as a prediction tool, selecting the best alternative(s) (voting), OCR, as well as software engineering. The fact that See5 can find useful patterns in sets of data has a lot of applications. In some sense you could call all these tasks "data mining" but it is not always what we typically mean by data mining. See5 is used for a wide variety of different kinds of tasks. See5 extensively from version 1.19 (2003) up until the latest update which is 2.08. However, See5 is not only used by Software Engineers but also by researchers, academia, and data mining professionals, students, small companies, and even hobbyists.See5 downloads for Windows PC (2000/XP/Vista/7) and Linux. See5 is very simple to use. It is a simple flat form based user interface with three menus "File", "Edit", and "Help" as well as five icons/buttons, which correspond to commonly menu items.

- Locate Data
- Construct Classifier
- Stop
- Review Output
- Use Classifier
- Cross-Reference

Before you use See5 you have to create at least two files, a ".data" file and a ".names" file. The data file is used for training, which is the same as the creation of the decision tree. It contains rows which consist of a data ID or a label, and then a so called feature/attribute vector, which is comma separated data, and this is followed by the correct "class" for this data. See5 is not as accurate as many non-linear classifiers, for example, neural networks, or polynomial classifiers, at least if you

have a large data set that can be represented numerically. The reason See5 are used instead of neural networks for many tasks are:

(1) Unlike neural networks, decision trees and rule sets are human readable, possible to comprehend, and can be modified manually if necessary. Since we often have problems with non-representative data but we typically understand these problems as well as our system quite well, it was sometimes advantageous for us to modify the decision trees directly. If you have questions about a certain decision that the classifier has made you can simply examine the tree and see how the decision occurred. Neural network algorithms produce matrixes of numbers, which cannot be modified manually or readily understood.

(2) We often have a lot of nominal data, which essentially is non-numerical data (colors, Yes/No, limited sets, etc). Some nominal data can be converted into a numerical representation, for example, when it is binary (Yes/No, 1/0), or when the possible attributes can be ordered numerically (short, medium, long, very long). Nominal data that cannot be converted into numerical format (and make sense), for example, colors, types of something, does not go well with neural networks.

(3) The same is true when you have unknown attributes. See5 can handle unknown attributes (denoted with a ?) but not neural networks and many other classifiers.

(4) See5 still performs decently when the data set is small (unlike many non-linear classifiers which needs larger training sets).

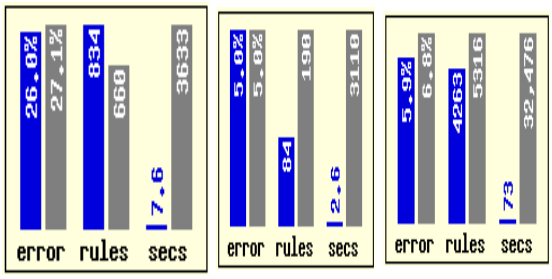
One great risk with See5 (or any classifier system including neural networks) is the risk for over fit. This means that you are matching to the data too much and you may be picking up statistical noise instead of true relations between the data. This may distort the tree so it makes poor predictions/classifications. The best way to monitor this problem is to create a blind set. A blind set is just like training data in the sense that you know the class (or the diagnosis). However, you use it as unknown data at the end to see how good your classifier actually is.

2.2 C5.0

Unix counterpart C5.0 are sophisticated data mining tools for discovering patterns that delineate categories, assembling them into classifiers, and using them to make predictions.C5.0 incorporates several new facilities such as variable misclassification costs. C5.0 allows a separate cost to be defined for each predicted/actual class pair; if this option is used, C5.0 then constructs classifiers to minimize expected misclassification costs rather than error rates. C5.0 has provision for a case weight attribute that quantifies the importance of each case; if this appears, C5.0 attempts to minimize the weighted predictive error rate. C5.0 has several new data types in addition to those available in C4.5, including dates, times, timestamps, ordered discrete attributes, and case labels. In addition to missing values, C5.0 allows values to be noted as not applicable. Further, C5.0 provides facilities for defining new attributes as functions of other attributes. Some recent data mining applications are characterized by very high dimensionality, with hundreds or even thousands of attributes. C5.0 can automatically winnow the attributes before a classifier is constructed, discarding those that appear to be only marginally relevant. For high-dimensional applications, winnowing can lead to smaller classifiers and higher predictive accuracy, and can often reduce the time required to generate rulesets. C5.0 is also easier to use.

So, let's see how C5.0 stacks up against C4.5.

Rulesets: often more accurate, much faster, and much less memory



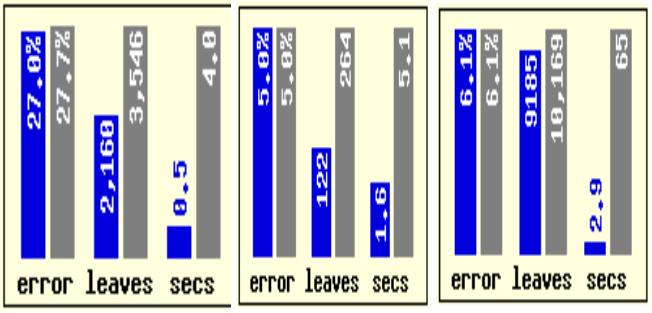
sleep ***income*** ***forest***

These graphs show the accuracy on unseen test cases, number of rules produced, and construction time for the three datasets. Results for C5.0 are shown in blue. Both C4.5 and C5.0 can produce classifiers expressed either as decision trees or rulesets. In many applications, rulesets are preferred because they are simpler and easier to understand than decision trees, but C4.5's ruleset methods are slow and memory-hungry. C5.0 embodies new algorithms for generating rulesets, and the improvement is substantial.

- ***Accuracy:*** The C5.0 rulesets have noticeably lower error rates on unseen cases for the ***sleep*** and ***forest*** datasets. The C4.5 and C5.0 rulesets have the same predictive accuracy for the ***income*** dataset, but the C5.0 ruleset is smaller.

- ***Speed:*** C5.0 is much faster; it uses different algorithms and is highly optimized. For instance, C4.5 required nine hours to find the ruleset for ***forest***, but C5.0 completed the task in 73 seconds.
- ***Memory:*** C5.0 commonly uses an order of magnitude less memory than C4.5 during ruleset construction. For the ***forest*** dataset, C4.5 needs more than 3GB (the job would not complete on earlier 32-bit systems), but C5.0 requires less than 200MB.

Decision trees: faster, smaller

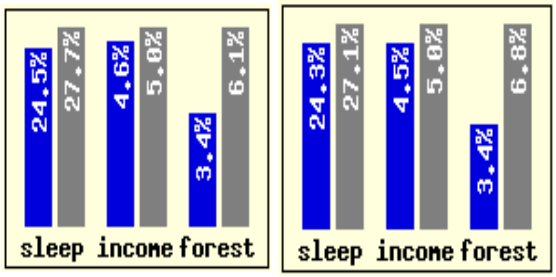


sleep ***income*** ***forest***

For all three datasets, C4.5 and C5.0 produce trees with similar predictive accuracies (although C5.0's is marginally better for the ***sleep*** application). The major differences are the tree sizes and computation times; C5.0's trees are noticeably smaller and C5.0 is faster by factors of 8, 3, and 22 respectively.

Boosting: !!!

Error rates on unseen test cases



boosted decision trees ***boosted rulesets***

Based on the research of Freund and Schapire, this is an exciting new development that has no counterpart in C4.5. Boosting is a technique for generating and combining multiple classifiers to improve predictive accuracy. The graphs above show what happens in *10-trial boosting* where ten separate decision trees or rulesets are combined to make predictions. The error rate on unseen cases is reduced for all three datasets, substantially so in the case of ***forest*** for which the error rate of boosted classifiers is about half that of the corresponding C4.5 classifier. Unfortunately, boosting doesn't always help -- when the training cases are noisy, boosting can actually reduce classification accuracy. C5.0 uses a novel variant of boosting that is less affected by noise, thereby partly overcoming this limitation. C5.0 supports boosting with any number of trials. Naturally, it takes longer to produce boosted classifiers, but the results can justify the additional computation! Boosting should always be tried when peak predictive accuracy is required, especially when unboosted classifiers are already quite accurate.

III. Testing and Evaluation

- See5/C5.0 has been designed to analyze **substantial databases** containing thousands to millions of records and tens to hundreds of numeric, time, date, or nominal fields. See5/C5.0 also takes advantage of computers with up to eight cores in one or more CPUs (including Intel Hyper-Threading) to speed up the analysis.
- To maximize interpretability, See5/C5.0 classifiers are expressed as **decision trees** or **sets of if-then rules**, forms that are generally easier to understand than neural networks.
- See5/C5.0 is available for **Windows Xp/Vista/7/8** and **Linux**.
- See5/C5.0 is **easy to use** and does not presume any special knowledge of Statistics or Machine Learning (although these don't hurt, either!)
- RuleQuest provides **C source code** so that classifiers constructed by See5/C5.0 can be embedded in your organization's own systems.

3.1 Sample Applications Using See5.

Detecting Advertisements on the Web

The example uses innovative data from Nick Kushmerick. There are 3279 cases, each describing an image within an anchor tag in a HTML document. About 14% of these anchored images are banner advertisements, and the goal is to generate rules that predict whether an image is an ad. (Kushmerick's system *AdEater* uses this prediction to eliminate advertisement images and so speed up page downloading.) This dataset is very high-dimensional -- there are 1558 attributes, about half the number of cases! These features include three numbers -- image height, width, and aspect ratio -- together with boolean features representing the presence or absence of phrases in the image caption, its *alt* tag, and the anchor, image, and base URLs. For example, the attribute **ancurl*http+www** has the value 1 if the URL referred to in the anchor contains *http* followed by *www* (ignoring punctuation). More than a quarter of the cases have unknown values for one or more of the attributes. In 0.7 seconds See5 extracts 16 rules from these data, 15 describing ads and one for non-ads. Some examples are:

Rule 1: (65, lift 7.0)
ancurl*http+www = 1
ancurl*click = 0
-> class ad [0.985]

Rule 4: (24, lift 6.9)
url*ad+gif = 0
ancurl*http+www = 0
ancurl*ad = 1
-> class ad [0.962]

Rule 16: (3127/313, lift 1.0)
url*ads = 0
-> class nonad [0.900]

On a ten-fold cross-validation,C5.0's rulesets correctly classify 97% of unseen cases, once again showing that simple theories can be useful too.

3.2 Sample Applications Using C5.0.

Diagnosing Hypothyroidism

The data for this example come from an assay screening service related to thyroid function and concern one aspect (hypothyroidism) of thyroid diagnosis. The attributes are a mixture of measured and calculated values and information obtained from the referring physician. There are four classes: negative, primary hypothyroid, secondary hypothyroid, and compensated hypothyroid. Let's show a few examples:

Attribute	***Assay 1***	***Assay 2***	***Assay 3***
age	32	63	19
sex	F	M	M
on thyroxine	t	f	f
query on thyroxine	f	f	f
on antithyroid medication	f	f	f
sick	f	f	f
pregnant	t	N/A	N/A
thyroid surgery	f	f	f
I131 treatment	f	f	f
query hypothyroid	f	f	t
query hyperthyroid	t	f	f
lithium	f	f	f
tumor	f	f	f
goitre	f	f	f
hypopituitary	f	f	f
psych	f	f	f
TSH	0.025	108	9
T3	3.7	.4	2.2
TT4	139	14	117
T4U	1.34	.98	-

FTI 104 14 -
referral source other SVI other
diagnosis negative primary compensated
hypothyr hypothyr

C5.0 processes 2,772 such cases in less than one-tenth of a second, giving seven rules for three of the classes. (The cases for the fourth class were too few in number to justify any rules.)

Rule 1: (31, lift 42.7)
thyroid surgery = f
TSH > 6
TT4 <= 37
-> class primary [0.970]

Rule 2: (63/6, lift 39.3)
TSH > 6
FTI <= 65
-> class primary [0.892]

Rule 3: (270/116, lift 10.3)
TSH > 6
-> class compensated [0.570]

Rule 4: (2225/2, lift 1.1)
TSH <= 6
-> class negative [0.999]

Rule 5: (296, lift 1.1)
on thyroxine = t
FTI > 65
-> class negative [0.997]

Rule 6: (240, lift 1.1)
TT4 > 153
-> class negative [0.996]

Rule 7: (29, lift 1.1)
thyroid surgery = t
FTI > 65
-> class negative [0.968]

Despite their simplicity, these rules are remarkably accurate -- their error rate on a further 1000 unseen test cases is only half of one percent!

3.3 Evaluation

Classifiers constructed by See5 are evaluated on the training data from which they were generated, and also on a separate file of unseen test cases if available; evaluation by cross-validation is discussed elsewhere. Results of the decision tree on the cases in hypothyroid data are:

Decision Tree

Size Errors
12 7(0.3%) <<

Size is the number of non-empty leaves on the tree and Errors shows the number and percentage of cases misclassified. The tree, with 12 leaves, misclassifies 7 of the 2772 given cases, an error rate of 0.3%. This might seem inconsistent with the errors recorded at the leaves -- the leaf mentioned above shows 24.8 errors! The discrepancy arises because parts of a case split as a result of unknown attribute values can be misclassified and yet, when the votes from all the parts are aggregated, the correct class can still be chosen. When there are no more than twenty classes, performance on the training cases is further analyzed in a *confusion matrix* that pinpoints the kinds of errors made.

(a) (b) (c) (d) <-classified as
---- ---- ---- ----
60 3 (a): class primary
153 1 (b): class compensated
2 (c): class secondary
1 2552 (d): class negative

In this example, the decision tree misclassifies.

- three of the primary cases as compensated,
- one of the compensated cases as negative,
- both secondary cases as negative, and
- one negative case as compensated.

When the number of classes is larger than twenty, a summary of performance broken down by class is shown instead. The entry for each class shows the number of cases for that class and the numbers of *false positives* and *false negatives*. A false positive for class *C* is a case of another class that is classified as *C*, while a false negative for *C* is a case of class *C* that is classified as some other class. Of course, the total number of errors must come to half the sum of the numbers of false positives and false negatives, since each error is counted twice--as a false negative for its true class, and as a false positive for the predicted class. For some applications, especially those with many attributes, it may be useful to know how the individual attributes contribute to the classifier. This is shown in the next section:

Attribute usage:

90% TSH
18% thyroid surgery
17% on thyroxine
14% TT4
13% T4U
13% FTI
7% referral source

The figure before each attribute is the percentage of training cases in hypothyroid.data for which the value of that attribute is known and is used in predicting a class. The second entry, for instance, shows that the decision tree uses a known value of thyroid surgery when classifying 18% of the training cases. Attributes for which this value is less than 1% are not shown. Two points are worth noting here:

- These values are computed for the particular classifier and training cases; changing either would give different values.
- When a case is classified, use of an attribute such as FTI that is defined by a formula also counts as using any attributes involved in its definition (here TT4 and T4U).

If there are optional unseen test cases, the classifier's performance on these cases is summarized in a format similar to that for the training cases.

```
        Decision Tree
      ----------------
      Size      Errors
        12    4( 0.4%)   <<
       (a)   (b)   (c)   (d)    <-classified as
      ----  ----  ----  ----
      31                 1    (a): class primary
           39                 (b): class compensated
                              (c): class secondary
           2          926    (d): class negative
```

A very simple *majority* classifier predicts that every new case belongs to the most common class in the training data. In this example, 2553 of the 2772 training cases belong to class negative so that a majority classifier would always opt for negative. The 1000 test cases from file hypothyroid test include 928 belonging to class negative, so a simple majority classifier would have an error rate of 7.2%. The decision tree has a lower error rate of 0.4% on the new cases, but notice that this is higher than its error rate on the training cases. The confusion matrix (or false positive/false negative summary if there are more than twenty classes) for the test cases again provides more details on correct and incorrect classifications.

IV. Results & Conclusions

The value of predictive patterns lies in their ability to make accurate predictions! It is difficult to judge the accuracy of a classifier by measuring how well it does on the cases used in its construction; the performance of the classifier on new cases is much more informative. (For instance, any number of gurus tell us about patterns that `explain' the rise/fall behavior

of the stock market in the past. Even though these patterns may appear plausible, they are only valuable to the extent that they make useful predictions about future rises and falls.) The third kind of file used by See5 consists of new test cases (e.g. hypothyroid.test) on which the classifier can be evaluated. This file is optional and, if used, has exactly the same format as the data file. Another optional file, the cases file (e.g. hypothyroid.cases), differs from a test file only in allowing the cases' classes to be unknown (`?'). The cases file is used primarily with the cross-referencing procedure and public source code, both of which are described later on under linking to other programs.

Costs file (optional)

The last kind of file, the costs file (e.g. hypothyroid.costs), is also optional and sets out differential misclassification costs. In some applications there is a much higher penalty for certain types of mistakes. In this application, a prediction that hypothyroidism is not present could be very costly if in fact it is. On the other hand, predicting incorrectly that a patient is hypothyroid may be a less serious error. See5 allows different misclassification costs to be associated with each combination of real class and predicted class. We will return to this topic near the end of the tutorial.

User Interface

It is difficult to see what is going on in an interface without actually using it. As a simple illustration, here is the main window of See5 after the hypothyroid application has been selected.

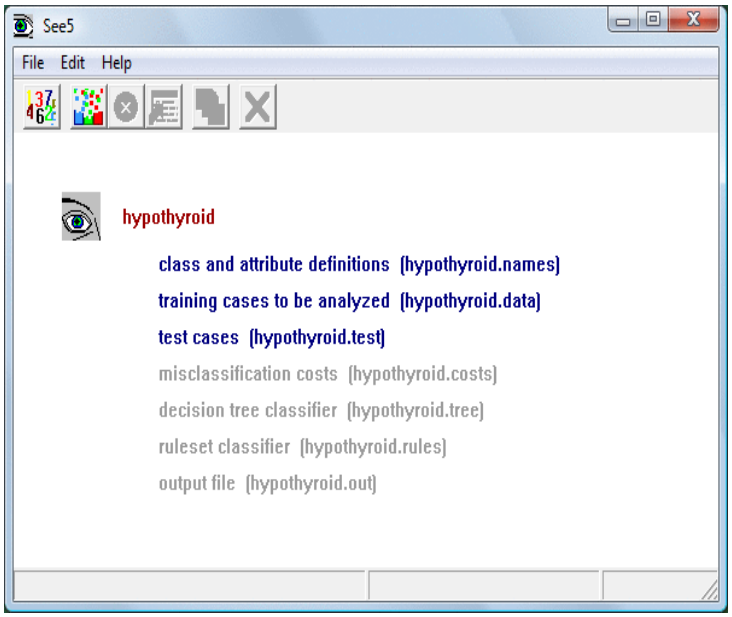


Fig. 1 See5 main window

The main window of See5 has six buttons on its toolbar. From left to right, they are

Locate Data: Invokes a browser to find the files for your application, or to change the current application.
Construct Classifier: selects the type of classifier to be constructed and sets other options.
Stop: Interrupts the classifier-generating process.
Review Output: re-displays the output from the last classifier construction (if any).
Use Classifier: interactively applies the current classifier to one or more cases and
Cross-Reference: shows how cases in training or test data relate to (parts of) a classifier and vice versa. These functions can also be initiated from the File menu. The Edit menu facilities changes to the names and costs files after an application's files have been located. On-line help is available through the Help menu.

Constructing Classifiers

Once the names,data and optional files have been set up, everything is ready to use See5. The first step is to locate the date using the Locate Data button on the toolbar (or the corresponding selection from the File menu). We will assume that the hypothyroid data above has been located in this manner. There are several options that affect the type of classifier that See5 produces and the way that it is constructed. The Construct Classifier button on the toolbar (or selection from the File menu) displays a dialog box that sets out these classifier construction options:

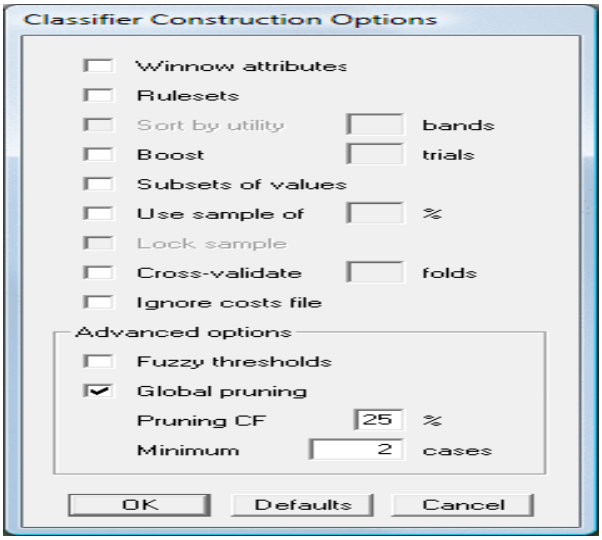


Fig. 2 Classifier Window

Many of the options have default values that should be satisfactory for most applications.

Decision trees

When See5 is invoked with the default values of all options, it constructs a decision tree and generates output like this: See5 [Release 2.10] Tue Mar 19 09:32:54 2013.

```
Class specified by attribute `diagnosis'
Read 2772 cases (24 attributes) from hypothyroid.data
Decision tree:
TSH <= 6: negative (2472/2)
TSH > 6:
:...FTI <= 65:
:...thyroid surgery = t:
:   :...FTI <= 36.1: negative (2.1)
:   :   FTI > 36.1: primary (2.1/0.1)
:   thyroid surgery = f:
:   :...TT4 <= 61: primary (51/3.7)
:       TT4 > 61:
:       :...referral source in {WEST,SVHD}: primary (0)
:           referral source in {STMW,SVHC,SVI}: primary (4.9/0.8)
:           referral source = other:
:           :...TSH <= 22: negative (6.4/2.7)
:               TSH > 22: primary (5.8/0.8)
FTI > 65:
:...on thyroxine = t: negative (37.7)
    on thyroxine = f:
:...thyroid surgery = t: negative (6.8)
    thyroid surgery = f:
:...TT4 > 153: negative (6/0.1)
    TT4 <= 153:
:...TT4 <= 37: primary (2.5/0.2)
    TT4 > 37: compensated (174.6/24.8)
Evaluation on training data (2772 cases):
          Decision Tree
        ----------------
        Size      Errors
          12    7( 0.3%)   <<
         (a)   (b)   (c)   (d)    <-classified as
```

```
        ----  ----  ----  ----
          60     3              (a): class primary
         153           1    (b): class compensated
                       2    (c): class secondary
           1        2552    (d): class negative
      Attribute usage:
         90%  TSH
         18%  thyroid surgery
         17%  on thyroxine
         14%  TT4
         13%  T4U
         13%  FTI
          7%  referral source
Evaluation on test data (1000 cases):
        Decision Tree
        ----------------
        Size      Errors
          12    4(0.4%)   <<
         (a)   (b)   (c)   (d)    <-classified as
        ----  ----  ----  ----
      31              1    (a): class primary
       1    39             (b): class compensated
                           (c): class secondary
         2         926    (d): class negative
Time: 0.0 secs
```

The first line identifies the version of See5 and the run date. See5 constructs a decision tree from the 2772 training cases in the file hypothyroid.data, and this appears next. Although it may not look much like a tree, this output can be paraphrased as:

if TSH is less than or equal to 6 then *negative*
else
if TSH is greater than 6 then
if FTI is less than or equal to 65 then
if thyroid surgery equals t then
if FTI is less than or equal to 36.1 then *negative*
else
if FTI is greater than 36.1 then *primary*
else
if thyroid surgery equals f then
if TT4 is less than or equal to 61 then *primary*
else
if TT4 is greater than 61 then
. . . .
and so on.

The tree employs a case's attribute values to map it to a *leaf* designating one of the classes. Every leaf of the tree is followed by a cryptic (*n*) or (*n*/*m*). For instance, the last leaf of the decision tree is compensated (174.6/24.8), for which *n* is 174.6 and *m* is 24.8. The value of *n* is the number of cases in the file hypothyroid.data that are mapped to this leaf, and *m* (if it appears) is the number of them that are classified incorrectly by the leaf. (A non-integral number of cases can arise because, when the value of an attribute in the tree is not known, See5 splits the case and sends a fraction down each branch.) The next section covers the evaluation of this decision tree shown in the second part of the output. Before we leave this output, though, its final line states the *elapsed* time for the run. (This differs from early releases of See5 which gave the CPU time.) The construction of a decision tree is usually completed quickly, even when there are many thousands of cases. Some of the options described later, such as ruleset generation and boosting, can slow things down considerably.

References:

[1] Comparison of C5.0 & CART Classification algorithms using pruning technique, "International Journal of Engineering Research & Technology" Vol. 1 Issue 4, June 2012,ISSN: 2278-0181.

[2] Shashi Shekhar, Pusheng Zhang, Yan Huang &Ranga Raju Vatsavai, *"Trends in Spatial Data Mining".*
[3] Yue He, Jingsi Liu, Lirui Lin *"Research on application of data mining"* School of Business and administration, Sichuan University May 2010.
[4] Zhu Xiaoliang, Wang Jian, Wu Shangzhuo & Yan Hongcan *"Research and Application of the improved Algorithm C4.5 on Decision Tree"* Hebei Polytechnic University, International Conference on Test and Measurement 2009.
[5] Mihai ANDRONIE, Daniel CRISAN Academy of Economic Studies, Bucharest, Romania Eidgenössische Technische Hochschule Zürich, *"Commercially Available Data Mining Tools used in the Economic Environment".*