Machine Learning: Implementing a Decision Tree Classifier

Motivation

To cement the concepts involved in the Decision Tree Classifier.

Big Picture

You will implement a Decision Tree Classifier. The data that you will work with is drawn from the UCI Machine Learning Repository. This is a repository of data that has been around since the mid 1980's for the express purpose of providing datasets on which individuals can apply and test machine learning techniques. The particular dataset upon which we will be working is the Breast Cancer Wisconsin (Original) Data Set.

Each record (row) includes 9 dimensions that describe nine attributes of benign and malignant tumors. While you do not need this information to complete the project, for those who are curious, the attributes are:

1. Clump Thickness
2. Uniformity of Cell Size
3. Uniformity of Cell Shape
4. Marginal Adhesion
5. Single Epithelial Cell Size
6. Bare Nuclei
7. Bland Chromatin
8. Normal Nucleoli
9. Mitoses

Each are measurements that have been transformed into integer values from 1 to 10. You can download the data here.

There is training data (train.csv) and test data (test.csv). The last column is the class column: 0 (benign) or 1 (malignant)

Your job will be to build a decision tree that can classify future (test) instances of tumor.

Specifics

You do not need to attempt any post-pruning or premature tree-growth-stoppage.

Create a README text file (or alternatively, a Word or RTF or PDF Document). You will submit your code and the README.

- It should contain your name and whether you are a grad student or not.
- It should describe how to run your program (how to compile if applicable, etc.).
- Include any detail on things like operating system or library requirements (like if you invoke any OS specific system-calls, use something like numpy, or need a statistics package included).
- Describe your approach to solving the programming problem (describe big-picture steps, how you organized your program, what methods you wrote, etc).
- The results for of your test run should be included.
- Include a list of what the root node question (attribute) was and what the next level questions are. You can draw this as a hierarchy if you wish.
- Describe how you handled "nonexistent answers". In other words, when classifying, did you just list a test instance with such an answer as unclassifiable or did you find a way to classify all instances?
- Describe any issues you weren't able to resolve and what you think the issue is.

RapidMiner

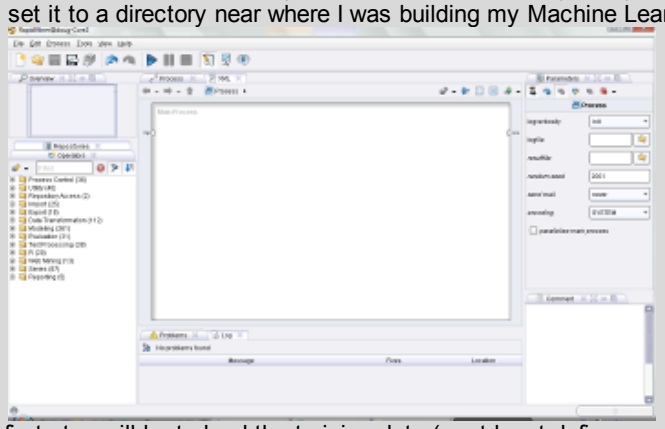
Next, you will get the opportunity to install and run some open source data mining software that has

Next, you will get the opportunity to install and run some open source data mining software that has recently received some popular press in the field. You will run its decision tree classifier on the same data and compare your results.

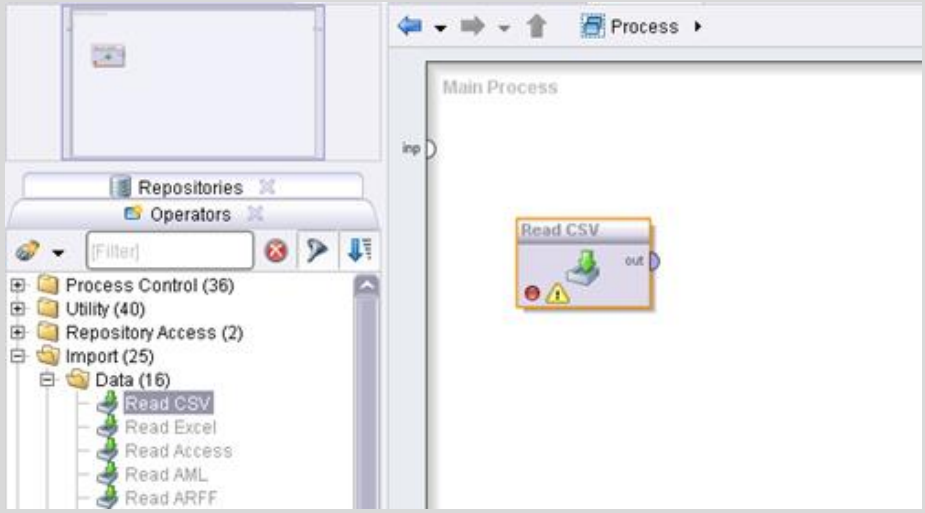
Installation and running instructions

You can find RapidMiner at http://rapid-i.com/content/view/181/190/

1. Go to the downloads pull-down and select RapidMiner
2. Download the Community Edition (found on the downloads page) and install it on your system.
3. Run the software. The first time you run the software it may ask you to set a Local Repository. I set it to a directory near where I was building my Machine Learning Projects.

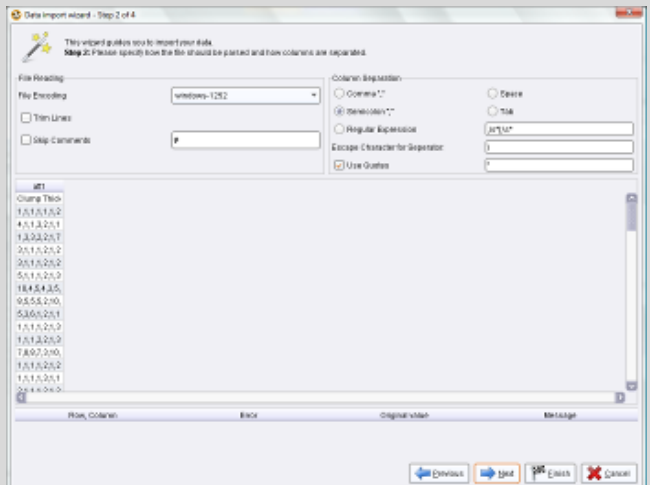


4. Your first step will be to load the training data (or at least define an operator that will do so when executed). Do this by dragging the "Read CSV" operator to the Main Process window (it is under Import,Data).

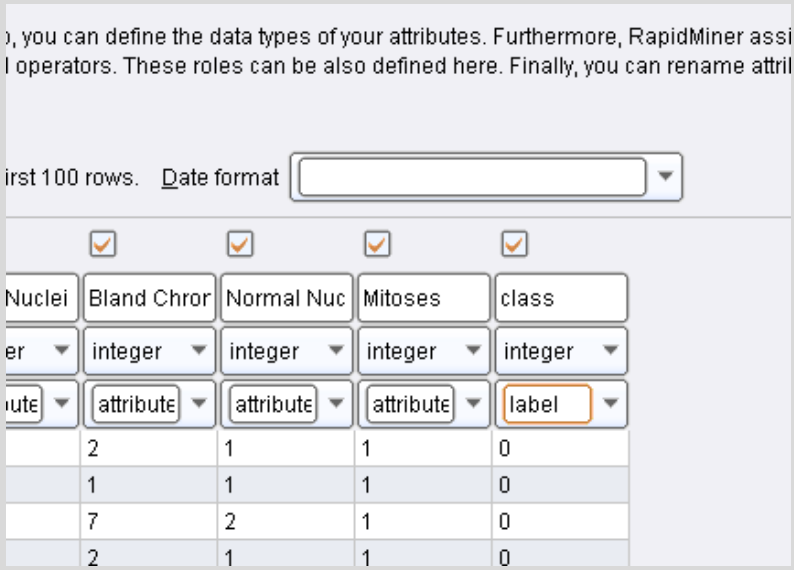


5. With it selected (highlighted as in above) click on the "Import Configuration Wizard" button over on the right hand side of the screen in the "Read CSV" parameter window.

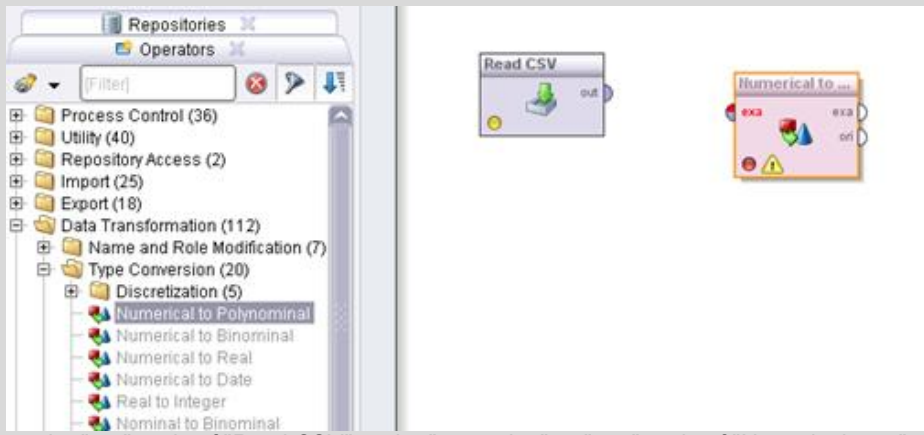
   Browse to where the training data is, select the train.csv file, and click next.



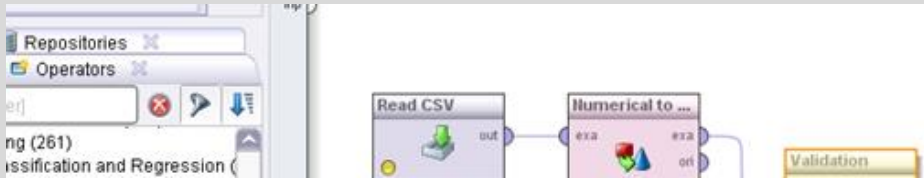
6. Select that column separation is caused by commas and select next. Press next again.
7. Note that the last column (class) is listed as an attribute. Change it to "label" and click "Finish".

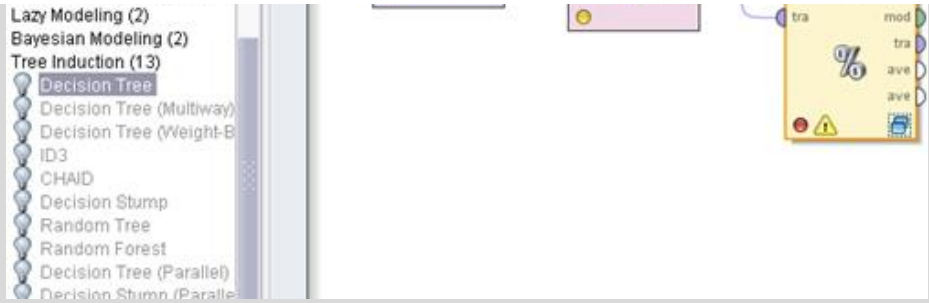


8. The decision tree in RapidMiner does not like integers as classes. It wants them to be nominal values (discrete) and does not easily interpret integers as such. We will bring in a "type conversion" operator to modify the data. Drag a "Numerical to Polynomial" operator to the Main Process window (under Data Transformation, Type Conversion).

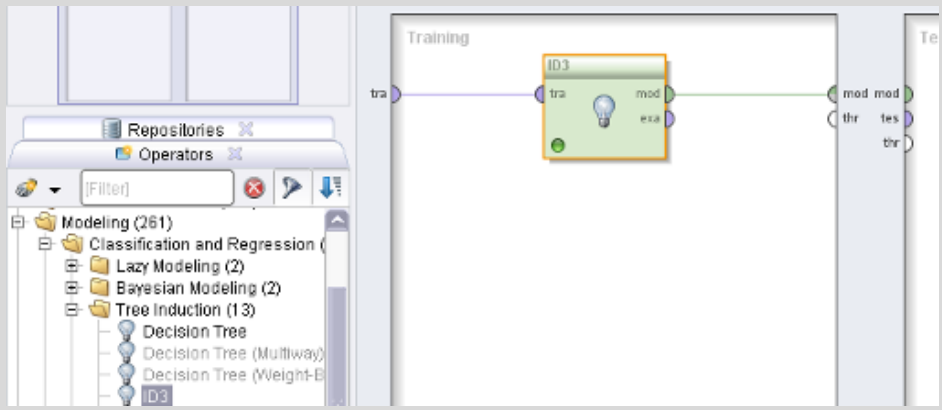


9. Connect the "out" node of "Read CSV" to the "examples" or "exa" node of "Numerical to…".
10. With "Numerical to…" highlighted set the "Attribute filter type" to all and click the "include special attributes" checkbox (on the right hand side in the "Numerical to Polynomial Parameters Pane."
11. Now we will bring our ID3 Classifier in, but it must be done within the context of Cross Validation. So drag an X-validation object into the Main Process pane (under Evaluation, validation). Connect the "exa" node on "Numerical to…" to the "Training" or "Tra" node on the X-validation object.

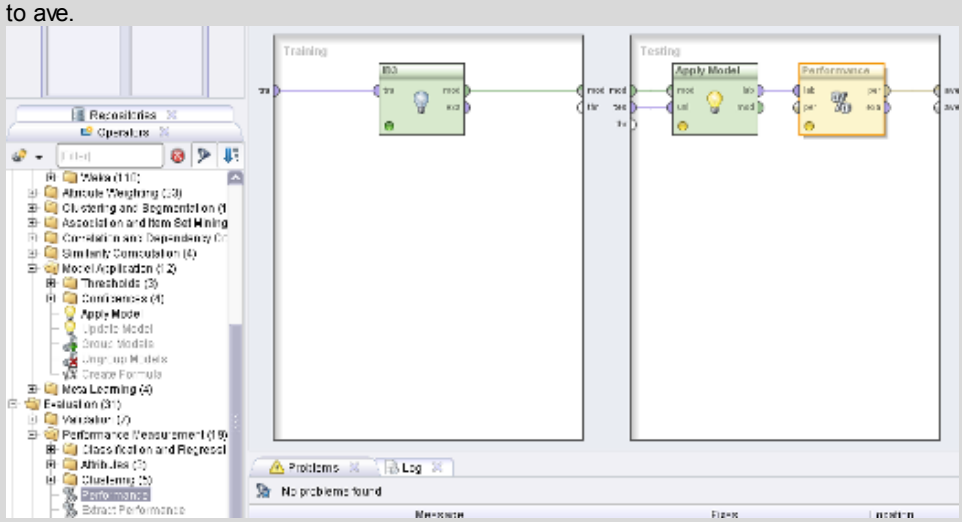




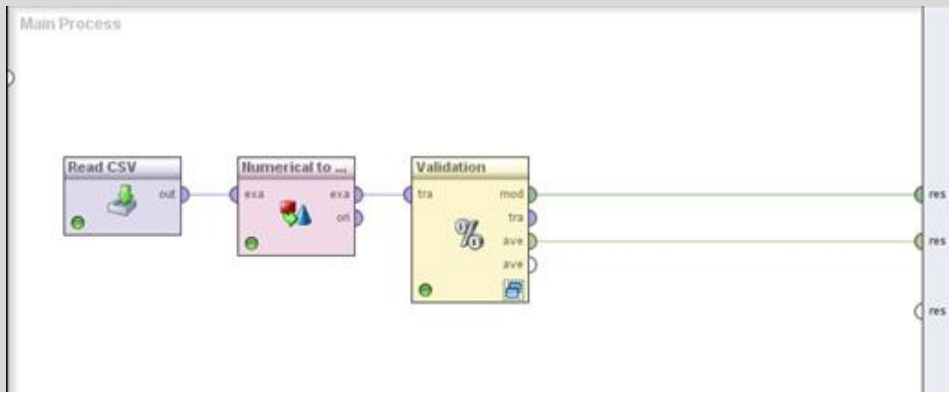
12. Double-click on it and drag an ID3 object into the training pane (under Modeling, Classification and Regression, Tree Induction). Connect the tra to tra nodes and connect the mod to mod nodes.



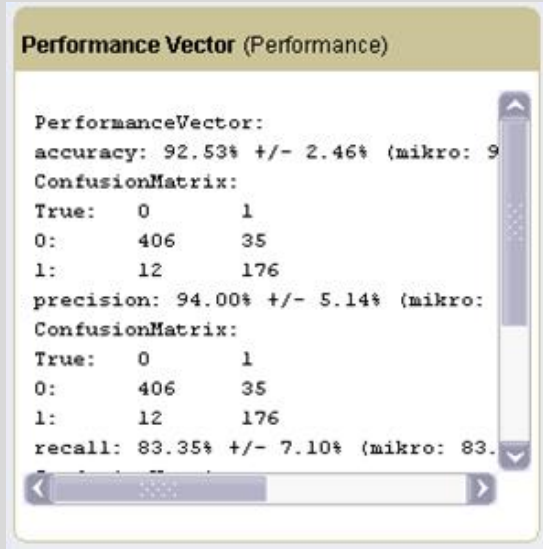
13. Now we must apply the trained model to the cross-validation data, so drag an "Apply Model" object to the test pane (under Modeling, Model Application). Connect mod to mod and tes to unl (unlabeled test data).
14. The output of this will be classified data so we need to measure how well the classifier does on this data. To do this drag a Performance object to the Test Pane. Connect lab to lab and per to ave.



15. Click on the up arrow above the training pane. Connect mod to res and ave to res (result) on the right of the Main Process Pane.

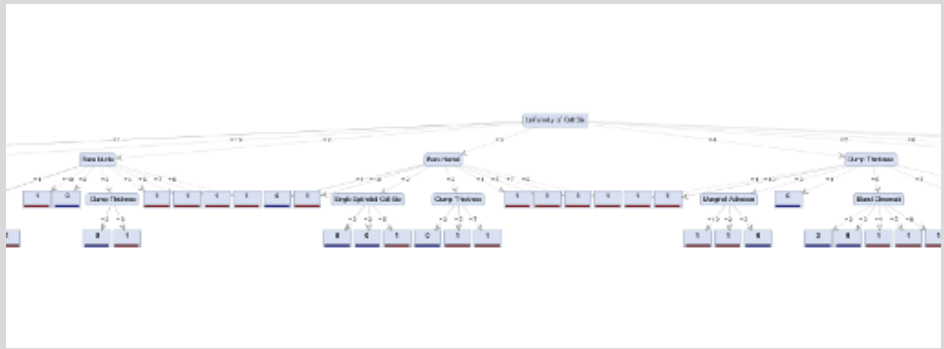


16. Click the play button at the top of the screen. It may prompt you to save the model (if you haven't been saving it all along. I saved the model in the repository created at the beginning). It will ask if you wish to switch to the results perspective. Answer yes.



Note that the performance of the ID3 10-fold cross-validation results is about 92.5% accurate, which is in the neighborhood of what we achieved with our implementation (note you might get something slightly different since this is affected by random chance and sampling).

17. Click on the "Tree(ID3)" tab while in the results perspective. You can get a graphical view of the tree.



18. Do a screen capture of the Main Process screen and of the derived tree. You can switch back to the design perspective by clicking the following button.

Tips

- When I loaded the data into Weka it reported 83% accuracy. This included 7 unclassifiable and 5 wrong. Mine did 90%, 7 wrong (out of 70), but mine was designed to classify everything.
- My implementation was recursive but that is not required.
- I didn't do anything fancy when advancing through possible values, just 1 to 10 (i.e. I didn't do them

- I didn't do anything fancy when advancing through possible values, just 1 to 10 (i.e. I didn't do them in order of their entropy).

Grade Rubric

D work:

- Source code, a README document, and two images (perhaps pasted into a document) have been submitted
- The README has the required information (see specifics above)
- The code reads in the training data, and at least an attempt is made to determine information gain for each dimension.

C work:

- All of the criteria of a D program must be met plus...
- Gains are determined, though perhaps incorrectly. Decision nodes are assembled, and the classifier accuracy beats what could be attained by random chance (just guessing should get you 50% accuracy).

B work:

- All of the criteria of a C program must be met plus...
- The code is well organized and easy to follow. This includes an adequate level of commenting so I can figure out what you are doing.
- Accuracy greater than 70%

A work:

- All of the criteria of a B program must be met plus…
- Your classifier operates at an accuracy level comparable to what I and others in the class achieve.

Zip your files into a single file for submission (source, readme file, and images).

Submission Instructions

1. Go to UM Online
2. Click on the link "Project submission: Implementing a Decision Tree Classifier (upload the assignment) (Due Sunday at 11:55 PM)"
3. Type any comments you wish.
4. Click on the Browse button to navigate to the directory containing your files for this project.
5. Highlight files you wish to upload.
6. If you have additional files to submit, repeat.
7. When you have finished adding all the files, click on **Send for grading**.