1. Introduction

This document describes how to use the CTree tool to build a decision tree and solve a classification problem using Excel.

About the tool

CTree is an Excel file with VBA code in the background to build and run the decision tree. CTree was created around 2001. It was released to public as a freeware through Yahoo Geocities. Over the years, quite a few people downloaded and used the tool for various purposes. They wrote back to the author with many feedbacks. Overall, the feedbacks have been positive. Few years back Yahoo Geocities was closed and the tool was no longer available. Author kept on receiving requests for the tool. This time, author is making it available again with minor modifications.

This tool is meant for solving classification problems with small to mid-sized data sets. The purpose is mainly to familiarize beginners with the classification problems, decision tree methodology, give them a light freeware to experiment on their own datasets. All they need is Microsoft Excel on their computers to get going.

This is not an industrial strength software and probably has many limitations and drawbacks. However in authors view, this is a good enough toy to explore decision trees, play around with different options, and study the impact on results and so on. In past people have used it as teaching aid in Statistics classes, used for solving problems from an amazing variety of domains like finance, geology, chemistry etc. One user even used this as an aid for betting on horses!

2. Dataset and the Problem

We will explain the tool with a specific classification problem. The data for the problem is publicly available from UC Irvine Machine Learning datasets repository (http://archive.ics.uci.edu/ml/datasets/Adult). The data is on ~32.5K individuals, their socio-demographic attributes and a flag whether their annual income was more than $50K or not. We will build a classification tree model to predict whether a person's annual income will be more than $50K or not given the same socio-demographic attributes of that person. We will assume that the reader is already familiar with the basics of Classification tree models.

The data set has 15 variables and ~32.5K rows. For the more detailed description of the dataset, problem and the variables, refer to the site (http://archive.ics.uci.edu/ml/datasets/Adult).

3. Data Preparation

As we have mentioned before, this tool has some limitations. That prevents user to use it on the full dataset. In this section we will describe some data preparation required before one submits the data to the tool.
Keep your data ready in an Excel (.xls) or CSV (.csv) file. Copy it from there to the Data worksheet of the tool. You should start pasting the data from the cell L23. The first row (row 23) should have the variable names and the following rows should be your data.

A few important tips at this point –

1. Paste your data as “values”, i.e. make sure you do not have Excel formulae in the data
2. Make sure that you do not have any blank row or blank column in the middle of the data
3. Your data may have missing values – make those cells blank, as opposed to using some other symbols

When we carry out this step for our data and try to run the tool, the first thing it complains about is the number of rows. Currently, the tool can handle maximum 10K rows. So we took ~10K rows from the top of our data set and pasted it in the appropriate lace of the Data worksheet.

Next step is to decide types of variables. Any variable (i.e. column) of your data can be one of the following four types.

a) Cont – Means, this variable should be used in the model as a predictor and it is a continuous variable. This is a numeric variable like Age, height, weight etc.
b) Cat - Means, this variable should be used in the model as a predictor and it is a categorical variable. Examples are – Occupation, color etc. Here the individual values are typically text strings and there is no "ordering" among the values (e.g. we cannot say RED is more than GREEN etc)
c) Class – This is a categorical variable. This is the "Class" variable of the classification problem. The aim of our model is to predict this variable based on the other variables. There should be exactly one class variable in the data.
d) Omit – Means, this variable, whatever it is, should NOT be used in the model.

You can specify the types by entering one of the 4 strings ("cont","cat","class","omit") in row 21. You should pick them from the in-cell drop-down menus. For our problem, ***Income*** is the class variable, ***Age*** is a "cont" variable, ***Region*** is a "cat" variable.

You might face a few bottlenecks here due to some restrictions imposed by the tool.

a) For a categorical variable, the number of unique categories should be less than 20. For example, if you have a variable (column) called Color and the values in that column are either RED or GREEN or BLUE, then the number of unique categories here is 3. In our example, the variable ***native-country*** had >20 unique values. So we created a new variable called ***Region*** out of that which has fewer categories. For example, whenever ***native-country*** was one of (Columbia, Ecuador, Nicaragua, Peru), we defined ***Region*** as "SouthAmerica". This is an example of how you may try to work around the problem.
b) For a categorical variable if you have a category with too few records (say 1 or 2 records), you need to either delete the row or change the category to some other existing category. In our example we had a single row with ***workclass*** ="Never-worked". We deleted that row from the data.
c) If you have too much missing data in a column, you should remove the column from modeling. In other words, you should set the type to 'Omit".

After few such initial hiccups, hopefully your data should be ready for building the model.

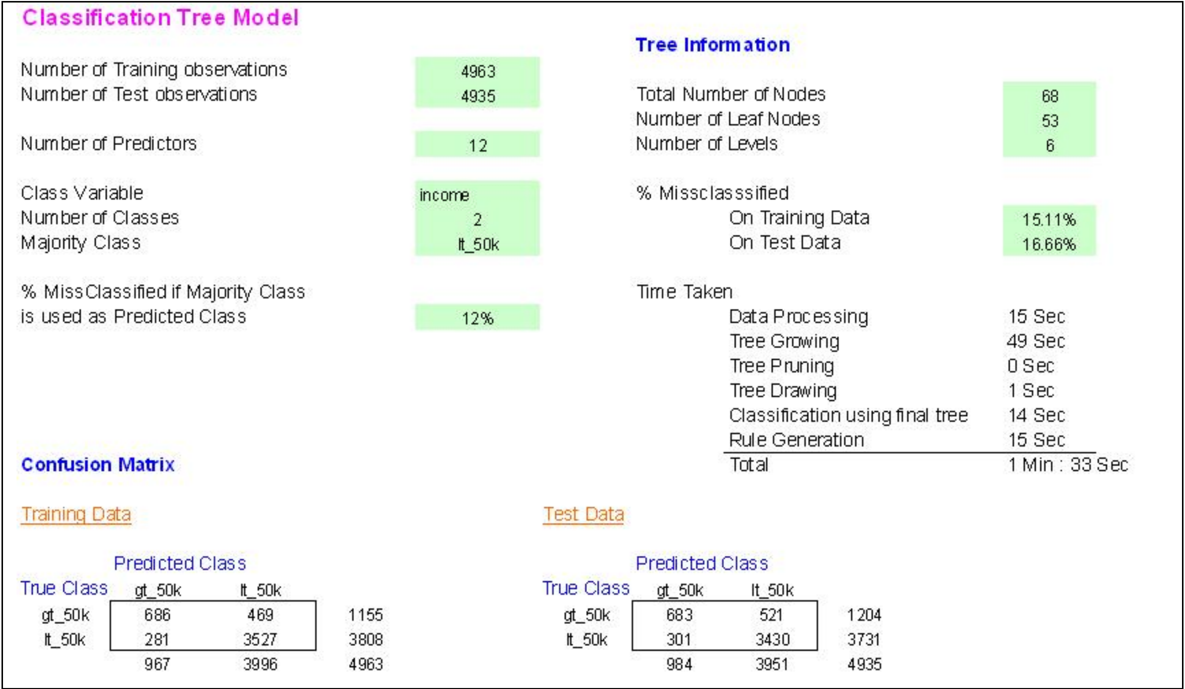
4. Building the Model

Once your data is ready, go to the ***UserInput*** tab to specify the model parameters. You may need to check various checkboxes and enter various options. We will not elaborate them here. The ***ReadMe*** worksheet and some background should help you there. After you are dome with specifying model parameters, hit the ***Build Tree*** button. This should start building the classification tree model.
At this point, the application will check various aspects of your input data. If it encounters any problem – will show you an error message and stop. At that point you will need to go back and fix the input data. If you have already taken care of the issues mentioned in the previous step, this step should go through smoothly without any error message. Depending on the size of your data (number of rows and number of variables) – it may take a while to finish building the model. To give an idea, for our example with ~10K rows and 13 variables, it takes about 1.5 – 2 minutes on a standard type of PC.

Once this step is over we are ready to explore the results.

5. Interpreting Results

You should look at the Results tab first.



Classification Tree Model

Number of Training observations	4963
Number of Test observations	4935
Number of Predictors	12
Class Variable	income
Number of Classes	2
Majority Class	lt_50k
% MissClassified if Majority Class is used as Predicted Class	12%

Tree Information

Total Number of Nodes	68
Number of Leaf Nodes	53
Number of Levels	6
% Missclasssified	
On Training Data	15.11%
On Test Data	16.66%
Time Taken	
Data Processing	15 Sec
Tree Growing	49 Sec
Tree Pruning	0 Sec
Tree Drawing	1 Sec
Classification using final tree	14 Sec
Rule Generation	15 Sec
Total	1 Min : 33 Sec

Confusion Matrix

Training Data

	Predicted Class		
True Class	gt_50k	lt_50k	
gt_50k	686	469	1155
lt_50k	281	3527	3808
	967	3996	4963

Test Data

	Predicted Class		
True Class	gt_50k	lt_50k	
gt_50k	683	521	1204
lt_50k	301	3430	3731
	984	3951	4935

Figure1: The Results tab showing the overall quality of the model

For the current problem we asked the tool to randomly select 50% of the data to build the model. This called Training data. The rest 50% is Test data, on which the model is evaluated. The results show that on training data about 15% of cases are misclassified (i.e. accuracy is ~85%). On test data, the misclassification rate is about 17%. This is actually not so bad. The UCI site from where the data is obtained, they report about 15% error rate for C4.5 algorithm (by the way, the is tool actually implements a version of C4.5). So we are in the same ballpark.

However, the situation is not very good either. Note that out of 2 categories (LT_50K,GT_50K) of the class variable ***Income***, LT_50K is the majority (i.e. there are more records with LT_50K than with GT_50K). The result sheet also reports that if we did not build any model and used a naïve rule – "No matter what, always predict the Income to be LT_50K ", then we would have made 12% error (less that what our sophisticated model is doing) !!!!

Where is the catch then? In the UCI site they got the 15% error rate on the whole data and not using some of the transformations (e.g. we used Region instead of native_country). Here we ran it on a SUBSET (that also, not a random subset) and had to take some shortcuts. After all that the tree model is not good. Anyway, I use this example to demonstrate the use of the tool – the showcased example is not necessarily the best use.

The Results page also breaks down the misclassification errors into two-types of errors and shows them in terms of confusion matrix. For example, on test data, there were 1204 records where the value of the class variable was "GT_50K" (i.e. annual income more than $50K). Out of these, our model got 683 cases right and for rest 521 cases; it mistakenly predicted that the class is "LT_50K" (i.e. annual income less than $50K).

Next, look at the Tree tab. This shows the tree model generated from the data

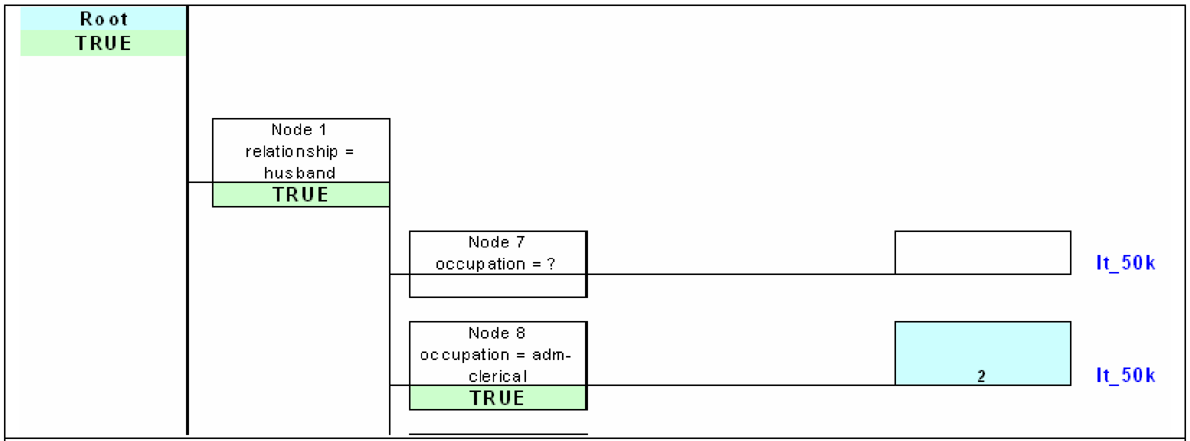


Figure2: Part of the tree model fitted to the data

For our example, the tree is pretty huge and we are showing only the top part of it. For example, it says that – IF relationship = "husband" AND occupation="adm-clerical" THEN the Class = LT_50K. Explore the other branches to study what the model is saying.

A nice feature of this page is that it can be used as a calculator. At the top-left side of the page in column G, the tool has listed the variables in the data. You can enter values of each variable next to it to see what the tree predicts for the observation you have entered. The path it takes in the tree is highlighted by a series of green cells with "TRUE" written inside them and the prediction is highlighted in blue. In above picture, it says that for the case we entered, relationship="husband" and occupation="adm-clerical" and hence the prediction is LT_50K.

In the NodeView page, you can see how your Training data is distributed among the nodes and what is the proportion of two classes (LT_50K : GT_50K) in each node. Notice that as you move towards the leaf nodes, this proportion gets more skewed to 1 (meaning you have nodes that contain predominantly one class).

CTree also generates rules based on the tree it constructed. See Rules page to view the rules. These are IF-THEN rules predicting the class. It also mentions the support and confidence of the rules. It is a rather long discussion to explain all the terminology here – so I am going to assume that the reader is already familiar with it. At a high level, you should look at rules that have high support and high confidence. For example, in our case it reported a rule - IF relationship=”unmarried” THEN income=”LT_50K”. This rule has 11% support and ~94% confidence. This means, in our training data, if a person is unmarried, 94% of the time his annual income is less than $50K. However, there are not a whole lot of unmarried persons in our data, there are about 11% of them. Hence, support of a rule says, how widely it is applicable and the confidence says, how accurate the rule is.

6. From here on …

This brief document hopefully gave you some idea on what you can do with CTree. Now you are on your own. Run it on different data sets, experiment with different options and see what you get. Comments are always welcome. You can reach me at adotsaha@yahoo.com

The earlier version of the tool had the macros locked with a password. About half of the requests I used to get were to unlock the application. Here it is. The code is now unlocked. Tools → Macros → Visual Basic Editor (Alt+F11) will open up the code for you.

Another set of comments I got earlier was related to this. Once you build the model, how do we run it on a new data set to actually predict? Unfortunately, the tool cannot do that and that is one major drawback.
HOWEVER!!! ….
You can use an option in UserInput sheet to save the tree model in a new worksheet. Then, if you setup your new data there and link things properly, you can get that feature. One example is provided in the TreeOutput.xls in the attachment. See if you can break the code.

I would really love to see that people improving the VBA code, making it faster, smarter, easier to understand. Since I cannot claim the best coding skills in the world, I believe there is plenty of room to improve this and make it available to enthusiasts to love to play with data. Enjoy!