INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Teaching Decision Tree Classification Using Microsoft Excel

Kaan Ataman, George Kulick, Thaddeus Sim,

Vol. 11, No. 3, May 2011, pp. 123–131
ISSN 1532-0545 | 11 | 1103 | 0123
INFORMS
Transactions on Education

informs®
DOI 10.1287/ited.1100.0060
© 2011 INFORMS

Teaching Decision Tree Classification Using Microsoft Excel

Kaan Ataman
Argyros School of Business and Economics, Chapman University, Orange, California 92866,
ataman@chapman.edu

George Kulick, Thaddeus Sim
Le Moyne College, Syracuse, New York 13214
{kulick@lemoyne.edu, simtk@lemoyne.edu}

Data mining is concerned with the extraction of useful patterns from data. With the collection, storage, and processing of data becoming easier and more affordable by the day, decision makers increasingly view data mining as an essential analytical tool. Unfortunately, data mining does not get as much attention in the OR/MS curriculum as other more popular areas such as linear programming and decision theory. In this paper, we discuss our experiences in teaching a popular data mining method (decision tree classification) in an undergraduate management science course, and we outline a procedure to implement the decision tree algorithm in Microsoft Excel.

Key words: data mining; decision tree classifier; spreadsheet modeling
History: Received: January 2010; accepted: August 2010.

1. Introduction

Advances in information technology systems and e-commerce over the past decade have allowed companies to collect enormous amounts of data on their business and customers (Babcock 2006). In recent years, companies have begun to explore whether useful information can be extracted from this data to be used to benefit their businesses. Netflix (2006), for instance, sponsored the Netflix Prize competition to help the company "improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences" to better meet their objective of "connect[ing] people to the movies they love."

In the context of data mining, this process of extracting information from raw data is known as knowledge discovery in databases (KDD). The KDD process includes data procurement, preprocessing of data, data mining, and interpretation of results (Tan et al. 2006). Our focus is on the data mining process, which is "the application of specific algorithms for extracting patterns from data" (Fayyad et al. 1996).

Data mining itself is not a new concept as evidenced by at least two decades' worth of research in the field. It has, however, not gained much traction in the OR/MS curriculum and it does not appear in many commonly used OR/MS textbooks such as Albright et al. (2008), Anderson et al. (2010), Hillier and Lieberman (2009), and Ragsdale (2007). Because data mining is concerned with the extraction of useful patterns from data to aid with decision making, it certainly falls into the field of OR/MS, which itself is involved with the use of analytical models to convert data into useful information for decision making. Thus, we believe that data mining should be part of any OR/MS curriculum and that a student's OR toolbox would be incomplete without exposure to it.

In this paper, we describe how we have incorporated data mining into an undergraduate elective management science course at a business school. This course is case-based and taught in a computer lab with an emphasis on spreadsheet modeling and problem solving. We cover four topics in the course: decision theory, revenue management, data mining, and optimization. For each topic, two to four 75-minute class sessions are devoted to basic theory and students then work on cases in groups. Selected student groups present their case work in subsequent classes. Students who take this course have already completed the introductory management science course that is required of all business students. The introductory course, which emphasizes problem solving and spreadsheet modeling skills, covers topics such as linear programming, Monte Carlo simulation, time-series forecasting, aggregate planning, and inventory management. With this foundation, students are able to tackle the more advanced material taught in our course.

Data mining is a broad field of study, and it is not possible to cover the entire field in about a quarter of a semester. Because students will likely see data mining techniques used for predictive purposes, we focus primarily on one such predictive technique called decision tree classification. Decision tree classification algorithms are covered in many data mining textbooks such as those by Witten and Frank (2005), Tan et al. (2006), and Olson and Shi (2007).

The contribution of our work is a self-contained teaching module that OR/MS educators can incorporate directly into or adapt for their own courses. Our use of Microsoft Excel to implement the decision tree algorithm eliminates the need to devote class time to teaching specialized data mining software. Also, the data set used in the case has issues commonly seen in real-life data such as missing or incomplete data. By having students go through the process of verifying and cleaning their data, they learn how to deal with missing or noisy data should they encounter this in the future.

The outline of the paper is as follows. A brief review of the data mining literature is provided in the next section. When teaching this data mining topic in our course, we begin by illustrating the decision tree classification algorithm using a simple example of assessing whether a loan applicant is likely to default on the loan. This example, adapted from Tan et al. (2006), and the decision tree algorithm are described in §3. In §4, we illustrate an implementation of the decision tree algorithm in Microsoft Excel. In §5, we discuss our experience of teaching this data mining topic in the management science elective course and provide details of the case study assigned to the students. Concluding remarks follow in §6.

2. Data Mining

Data mining is concerned with the extraction of useful patterns from data. The identification of patterns can be performed in an ad hoc manner if the number of records (or entries) in the database and the number of fields (or attributes) per record is small. However, with many practical databases such as point-of-sales data, Web logs, and e-commerce data containing millions of records with hundreds of attributes (Fayyad et al. 1996, Witten and Frank 2005), a more systematic approach is needed.

Generally speaking, data mining tasks are predictive (identifying patterns for predictive purposes), explanatory (identifying patterns to help explain relationships in the data), or both. Classification, which is a predictive task, looks at assigning objects to one of several predefined categories or class values. Common applications of classification algorithms include categorization of customers as loyal or risky based on the recorded historical behavior (Wei and Chiu 2002), detection of spam e-mail messages based on the message header and content (Pantel and Lin 1998), categorization of cells as malignant or benign based on the results of various tests (Mangasarian et al. 1995), and classification of galaxies based on their shapes (Bershady et al. 2000).

Well-known classification algorithms include decision trees (Quinlan 1986), artificial neural networks (Rosenblatt 1958, Rumelhart et al. 1986), naive Bayes classifier (Domingos and Pazzani 1997), nearest neighbor algorithms (Dasarathy 1990), and support vector machines (Vapnik 1995).

3. Decision Tree Induction Algorithm

The basic concept behind the decision tree classification algorithm is the partitioning of records into "purer" subsets of records based on the attribute values. A pure subset is one in which all the records have the same class label. The end result of the decision tree algorithm is the output of classification rules that are simple to understand and interpret. This interpretability property is strength of the decision tree algorithms.

In general, these algorithms find the attribute that best splits a set of records into a collection of subsets with the greatest overall purity measure. The purity of a subset can be quantified by entropy, which measures the amount of information loss. Entropy is a real number between zero and one, where an entropy value of zero indicates that the data set is perfectly classified while a value of one indicates that no information has been gained. The algorithm recursively operates on each newly generated subset to find the next attribute with which to split the data set. The algorithm stops when all subsets are pure or some other stopping criterion has been met. Examples of stopping criteria include the exhaustion of attributes with which to split the impure nodes, predefined node size, and minimum purity level.

To illustrate the decision tree algorithm, we use a database of previous loan borrowers adapted from Tan et al. (2006). The data set, shown in Table 1, contains 10 records and each record has 4 attributes: home owner, marital status, annual income, and defaulted. The defaulted attribute is the class or prediction variable and it has two labels: yes and no. Three borrowers defaulted on their loans while the remaining seven did not. The possible values for the other three attributes are yes or no for home owner; single, married, or divorced for marital status; and

Table 1 Data Set of Previous Loan Borrowers

ID	Homeowner	Marital status	Annual income	Defaulted
1	Yes	Single	High	No
2	No	Married	Average	No
3	No	Single	Low	No
4	Yes	Married	High	No
5	No	Divorced	Average	Yes
6	No	Married	Low	No
7	Yes	Divorced	High	No
8	No	Single	Average	Yes
9	No	Married	Low	No
10	No	Single	Average	Yes

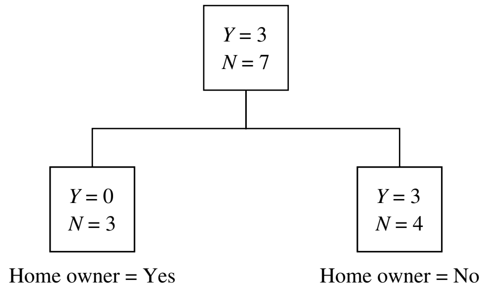
low, average, or high for annual income. The class variable defaulted is a binary nominal variable and so is the home owner attribute. Marital status is a nominal variable while annual income is ordinal.

As an aside, the annual income attribute in the original data set in Tan et al. (2006) is treated as a continuous variable with values ranging from 60 to 220. Decision trees require the evaluated attributes to be discrete rather than continuous. The discretization of a continuous variable adds another level of complexity to the tree-building process. For the purpose of this example, we eliminate this additional complexity by discretizing the continuous annual income attribute and assigning to it the value of low if the annual income is below the 25th percentile of this data set, average if the annual income is between the 25th and 75th percentiles, or high if the annual income is above the 75th percentile. Generally speaking, continuous attributes are discretized by evaluating several cutoff points to determine which cutoff point maximizes the information gain on that attribute. For a more detailed discussion of this discretization process, the reader is directed to Tan et al. (2006, p. 162).

The decision tree is constructed as follows. At each node of the tree, if the node is not pure (i.e., the records in the node do not all have the same class label), we split the node using the attribute that splits this parent node into a collection of child nodes with the greatest overall purity measure. For the loan data set, suppose we choose to split the data set using the home owner attribute at the top of the decision tree as shown in Figure 1. This results in two subsets or child nodes, one containing all records with home owner = yes and the other containing all records with home owner = no. In the home owner = yes child node, all three records have the class label defaulted = no. This is a pure subset because it contains only a single class label. On the other hand, in the home owner = no child node, three of the records have the class label defaulted = yes and four have defaulted = no. This subset is not pure.

The degree of impurity of a split is a weighted average of the impurity of the child nodes. A commonly

Figure 1 Splitting on the Home Owner Attribute



used measure for the impurity of the child node is entropy, which is calculated using the formula

$$\text{Entropy}(s) = -\sum_{i=0}^{c-1} p(i \mid s)\log_2 p(i \mid s), \tag{1}$$

where s is the value of the attribute used to split the parent node, $p(i \mid s)$ is the fraction of records belonging to class i using split s, and c is the number of classes. When calculating entropy, $0 \log_2 0$ is defined as zero. The $\log_2$ measure is used in binary classification because this represents the number of bits needed to specify the class in which a random instance belongs. The entropy function attains its maximum value at $p(i \mid s) = 0.5$, which represents an unbiased bit. This is the point where the child node is most impure. The entropy Equation (1) returns a value between zero and one, with zero indicating a pure child node. For further discussion on entropy and its use in information technology, see MacKay (2003).

Letting n_s be the number of records in the child node with attribute value s, S be the number of classes within the attribute, and N be the total number of records in the parent node, the degree of impurity of split s is

$$\text{Impurity} = \sum_{s=1}^{S} \frac{n_s}{N}\,\text{entropy}(s). \tag{2}$$

In our loan default problem, the number of class labels c is two. We will let $i = 0$ represent defaulted = no and $i = 1$ to represent defaulted = yes. If we were to split the parent node using the home owner attribute, the entropy values of the two child nodes are

$$\begin{aligned}
&\text{Entropy(home owner = yes)}\\
&\quad = -\left[\left(\tfrac{3}{3}\right)\log_2\left(\tfrac{3}{3}\right) + \left(\tfrac{0}{3}\right)\log_2\left(\tfrac{0}{3}\right)\right] = 0,\\
&\text{Entropy(home owner = no)}\\
&\quad = -\left[\left(\tfrac{4}{7}\right)\log_2\left(\tfrac{4}{7}\right) + \left(\tfrac{3}{7}\right)\log_2\left(\tfrac{3}{7}\right)\right]\\
&\quad = -[-0.461 - 0.524] = 0.985.
\end{aligned}$$

By weighing the entropy of each child node by the number of records in each child node relative to the total number of records in the parent node, we obtain the impurity measure for that particular split. Using Equation (2), the impurity value of the split using the home owner attribute is

$$\begin{aligned}&\tfrac{3}{10}\times \text{Entropy}(\text{home owner}=\text{yes})+\tfrac{7}{10}\\&\quad\times \text{Entropy}(\text{home owner}=\text{no})\\&=\tfrac{3}{10}\times 0+\tfrac{7}{10}\times 0.985=0.69. \qquad (3)\end{aligned}$$

Following the same procedure, the impurity values from splitting using the marital status and annual income attributes can be calculated. The resulting impurity values are 0.6 and 0.325, respectively. Because the annual income attribute has the lowest impurity value of the three attributes, this is the best attribute with which to split the root node of the decision tree. This is shown in Figure 2.

When splitting on the annual income attribute, the entropy values of the child nodes are

$$\text{Entropy}(\text{Annual Income}=\text{Low})=0,$$
$$\text{Entropy}(\text{Annual Income}=\text{Average})=0.81,$$
$$\text{Entropy}(\text{Annual Income}=\text{High})=0.$$

The annual income = low and annual income = high child nodes are pure, with all records in both nodes having a defaulted = no class label. No further processing of these two nodes is required. The child node annual income = average has three records with class label defaulted = yes and one record with class label defaulted = no. Because this node is not pure, we continue to split this node to obtain purer child nodes. Splitting this child node using the home owner attribute gives an impurity value of 0.811, while splitting on the marital status attribute gives an impurity value of zero. Thus, the best split is by marital status attribute. At this point, all of the end nodes in the decision tree are pure and the decision tree is complete.

The final decision tree for the loan default problem can be summarized pictorially as in Figure 3 or by

Figure 2 Splitting on the Average Income Attribute

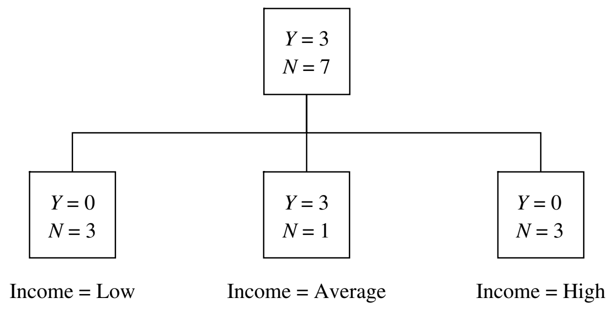
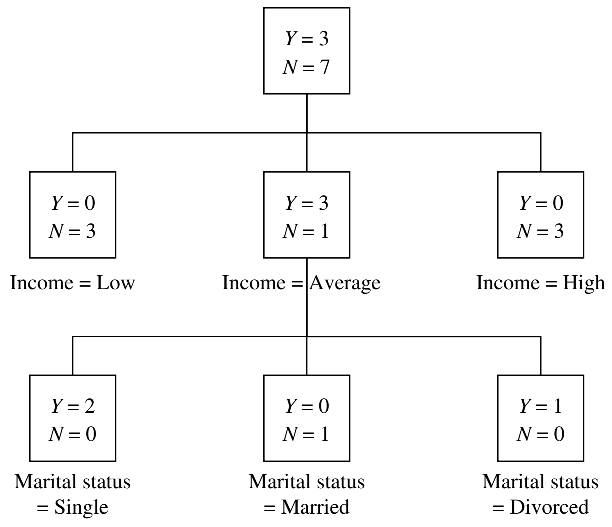


Figure 3 Completed Decision Tree



the following rules, which can be used in an expert system:

If the customer's annual income is either low or high, then the customer will not default on the loan.

If the customer's annual income is average and the customer is married, then the customer will not default on the loan.

Else the customer will default on the loan.

The decision tree algorithm described above is a version of the ID3 algorithm that, together with the C45 algorithm, is one of the two most commonly known decision tree algorithms (Quinlan 1986). The ID3 algorithm is desirable for its simplicity but it has several important drawbacks. For example, ID3 is a greedy search algorithm that picks the best attribute and does not reconsider earlier choices. This can often lead to problems especially when the data set contains noisy data, for example, two records containing similar attribute values but different class labels. Continuous splitting of the data set will never yield a pure subset and the tree will grow too large trying to explain a simple noise in the data. This issue can be mitigated through pruning, where a whole subtree is replaced by a leaf node (i.e., a node that does not have any child nodes) if the expected error rate of the subtree is greater than that of the single leaf node.

The pruning option does not exist in the ID3 algorithm but is available in the C45 algorithm (which builds on the ID3 algorithm). In addition, the C45 algorithm can handle missing or continuous variables (which the ID3 algorithm does not), has flexibility in the selection of alternative attributes (for instance, using attributes based on different costs or importance levels), and has improved computational efficiency.

4. Implementing the Decision Tree in Excel

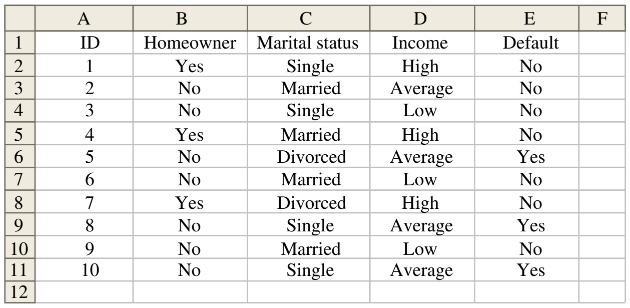
A significant portion of the work in building a decision tree is in the calculations of the impurity values for each possible split at each nonpure node in the tree. There are many software packages that automatically perform these calculations, such as the open-source software package Weka (Witten and Frank 2005) that is available from http://www.cs.waikato.ac.nz/~ml/. However, it is pedagogically instructive for students to manually build the decision tree to better understand the mechanics of the algorithm and the issues that may arise when constructing the decision tree.

Our choice of Microsoft Excel for this exercise is primarily because of its ability to quickly perform complex calculations. Once students understand how entropy and impurity are calculated, using Excel to perform these calculations will free them from this mechanical process so they may focus more on the structure of the tree. We will discuss this in further detail in §5. In addition, Excel allows us to work on larger data sets, which brings more realism to the topic while keeping the problem tractable without being too encumbered by the entropy and impurity calculations.

The number of entropy and impurity calculations remains an issue with the decision tree algorithm. In the worst case, the number of entropy values that have to be calculated is of the order $O(an^a)$, where a is the number of attributes in the data set, and $n = \max_{i=1,\ldots,a}\{n_i\}$, where n_i is the number of classes in attribute i. This problem is similar to the curse of dimensionality issue with the implementation of dynamic programming in Excel (Raffensperger and Pascal 2005). Though the issue of the exponential number of impurity calculations has yet to be resolved, we have designed an implementation procedure for problem sets with binary class variables that requires the modeler to perform several copy-and-paste operations and needs only some subsequent minor modifications to the pasted cells.

Figure 4 shows the loan data set in Excel, with the annual salary attribute converted into a nominal variable. We have this data located in a sheet titled "Data."

Figure 4 **Loan Data Set in Excel**



	A	B	C	D	E	F
1	ID	Homeowner	Marital status	Income	Default	
2	1	Yes	Single	High	No	
3	2	No	Married	Average	No	
4	3	No	Single	Low	No	
5	4	Yes	Married	High	No	
6	5	No	Divorced	Average	Yes	
7	6	No	Married	Low	No	
8	7	Yes	Divorced	High	No	
9	8	No	Single	Average	Yes	
10	9	No	Married	Low	No	
11	10	No	Single	Average	Yes	
12						

Figure 5 shows our implementation of the first level of the decision tree (see the sheet labeled Level 1). At the first level, the algorithm evaluates the splitting of the root node using each of the three attributes. The top table in Figure 5 corresponds to splitting the root node using the home owner attribute, the middle table to splitting using the marital status attribute, and the bottom table to the income attribute.

The table within Figure 5 illustrates that the split of the root node using the home owner attribute contains two pairs of rows, each row representing the scenario where the home owner attribute takes on the value of yes or no. The formulas in Row 4 of Figure 5 extract the number of records containing home owner = yes and calculates the entropy value for this child node. The formulas in Cells F4 to K4 are as below.

Cell	Formula	Copy to
F4	=DCOUNT(Data!\$A\$1:\$E\$11,"ID,"A3:D4)	G4
H4	=SUM(F4:G4)	
I4	=IF(F4=0,0,(F4/\$H4) ∗LOG(F4/\$H4,2))	J4
K4	=−SUM(I4:J4)	

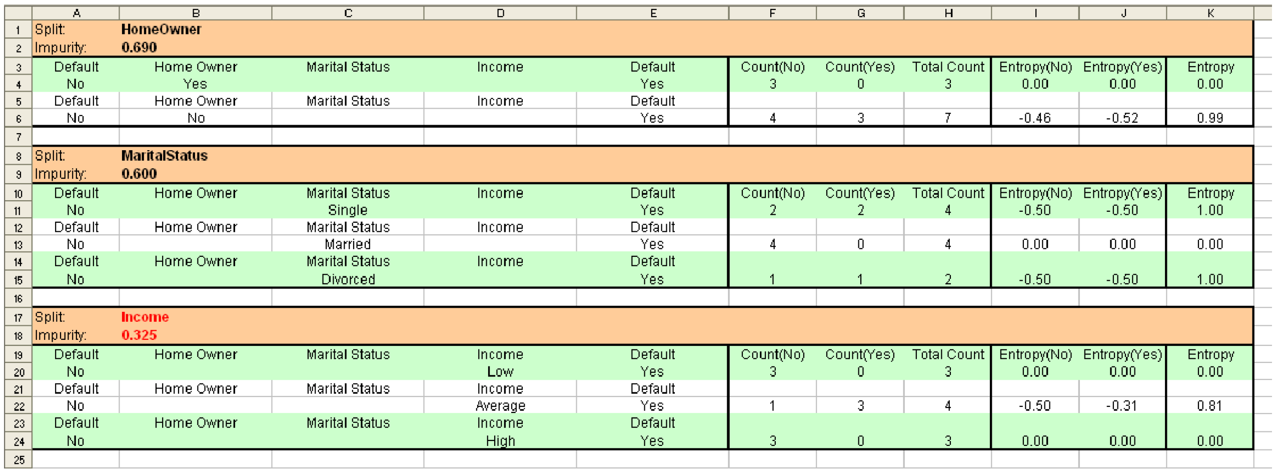
The DCOUNT formula is a query function that counts the number of records in the database (in this case, the loan data set table) that match the criteria default = no, home owner = yes, marital status = , income = . The DCOUNT formula ignores attributes that have blank values (i.e., marital status and income). When the formula in Cell F4 is copied to Cell G4, the DCOUNT formula in Cell G4 is updated to contain the default = yes criterion and drops the default = no criterion. The IF function in Cells I4 and J4 is used to return a value of zero instead of an error value when calculating $\log_2 0$. (Recall that we define $0 \log_2 0 = 0$.) The formula in Cell K4 is Equation (1).

The formulas in Row 6 of Figure 5 perform the same calculations for the home owner = no child node. Finally, in cell B2, we calculate the impurity value of this split (see Equation (3)) using the formula

$$= \mathrm{SUMPRODUCT}(\mathrm{H4{:}\ H6}, \mathrm{K4{:}\ K6})/\mathrm{SUM}(\mathrm{H4{:}\ H6}).$$

When one of the tables shown in Figure 5 is completed, one can create a copy of the table to evaluate other splits. For example, to evaluate splitting the root node using the marital status attribute, we copy the home owner table and make the necessary changes to the criteria used in the DCOUNT formula by leaving the home owner cells blank and entering the different

Figure 5 **Calculating the Impurity Values of Splitting the Root Node Using the Home Owner, Marital Status, and Income Attributes**



	A	B	C	D	E	F	G	H	I	J	K
1	Split:	**HomeOwner**									
2	Impurity:	**0.690**									
3	Default	Home Owner	Marital Status	Income	Default	Count(No)	Count(Yes)	Total Count	Entropy(No)	Entropy(Yes)	Entropy
4	No	Yes			Yes	3	0	3	0.00	0.00	0.00
5	Default	Home Owner	Marital Status	Income	Default						
6	No	No			Yes	4	3	7	-0.46	-0.52	0.99
7											
8	Split:	**MaritalStatus**									
9	Impurity:	**0.600**									
10	Default	Home Owner	Marital Status	Income	Default	Count(No)	Count(Yes)	Total Count	Entropy(No)	Entropy(Yes)	Entropy
11	No		Single		Yes	2	2	4	-0.50	-0.50	1.00
12	Default	Home Owner	Marital Status	Income	Default						
13	No		Married		Yes	4	0	4	0.00	0.00	0.00
14	Default	Home Owner	Marital Status	Income	Default						
15	No		Divorced		Yes	1	1	2	-0.50	-0.50	1.00
16											
17	Split:	**Income**									
18	Impurity:	**0.325**									
19	Default	Home Owner	Marital Status	Income	Default	Count(No)	Count(Yes)	Total Count	Entropy(No)	Entropy(Yes)	Entropy
20	No			Low	Yes	3	0	3	0.00	0.00	0.00
21	Default	Home Owner	Marital Status	Income	Default						
22	No			Average	Yes	1	3	4	-0.50	-0.31	0.81
23	Default	Home Owner	Marital Status	Income	Default						
24	No			High	Yes	3	0	3	0.00	0.00	0.00
25											

class values for the marital status attribute. As shown in Figure 5, an additional row has to be added to the table because the marital status attribute has three possible values: single, married, and divorced. The SUMPRODUCT impurity formula must be updated to include any newly added pairs of rows.

Figure 5 shows that splitting the root node by annual income provides the lowest impurity value with only the income = average child node being an impure node. We can use the same setup as before to split this node. Consider splitting the income = average node by the home owner attribute. As a shortcut, we can use a copy of the home owner table from the Level 1 sheet. In that table, we simply set income = average and the formulas will automatically recalculate to provide the impurity value of this split. Figure 6 shows the splitting of the income = average node by the home owner and marital status attributes.

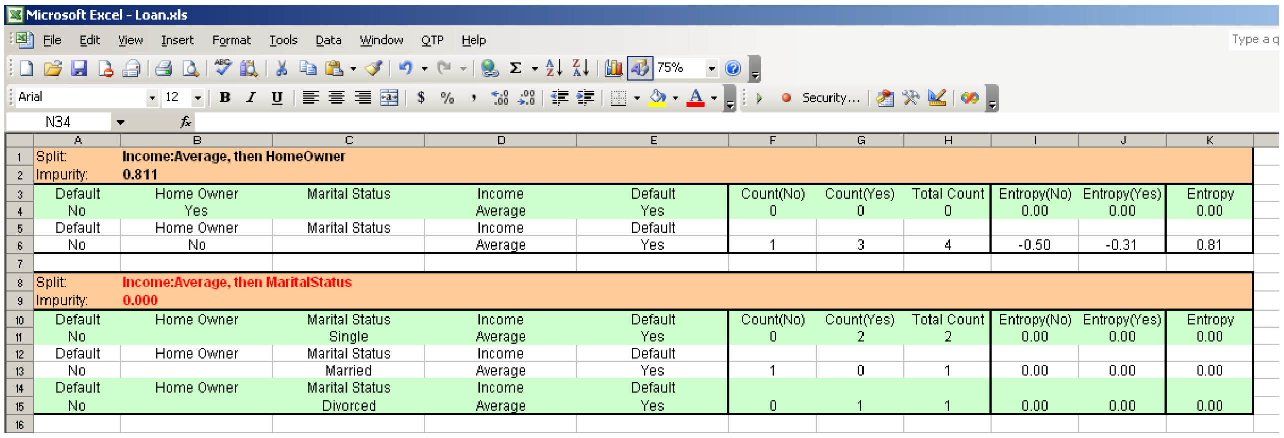
5. Classroom Experience

We have taught this material in an undergraduate management science elective course every year since 2008. The class met twice a week for 75 minutes each class period. The students were business majors of junior or senior standing, and they would have already taken the introductory management science course.

In this course, we cover four topics: decision theory, revenue management, data mining, and mathematical programming. For each topic, we typically spend two to four 75-minute class meetings discussing basic theory and working through examples, one class period outlining the details and setting expectations for the case the students will complete in groups, and one class period for the student groups to present their work on the case and the instructor to summarize the topic.

For the data mining module, we spent two class periods motivating the need to understand data mining, working through the bank loan example presented in §3 that includes calculating the entropy and impurity values by hand, implementing the decision tree induction algorithm in Microsoft Excel (see §4), and discussing some implementation issues with the decision tree induction algorithm.

Figure 6 **Calculating the Impurity Values of Splitting the Income = Average Node Using the Home Owner, and Marital Status Attributes**


Microsoft Excel - Loan.xls
File Edit View Insert Format Tools Data Window QTP Help
Type a q
75%
Arial
12
Security...
N34
fx

	A	B	C	D	E	F	G	H	I	J	K
1	Split:	**Income:Average, then HomeOwner**									
2	Impurity:	**0.811**									
3	Default	Home Owner	Marital Status	Income	Default	Count(No)	Count(Yes)	Total Count	Entropy(No)	Entropy(Yes)	Entropy
4	No	Yes		Average	Yes	0	0	0	0.00	0.00	0.00
5	Default	Home Owner	Marital Status	Income	Default						
6	No	No		Average	Yes	1	3	4	-0.50	-0.31	0.81
7											
8	Split:	**Income:Average, then MaritalStatus**									
9	Impurity:	**0.000**									
10	Default	Home Owner	Marital Status	Income	Default	Count(No)	Count(Yes)	Total Count	Entropy(No)	Entropy(Yes)	Entropy
11	No		Single	Average	Yes	0	2	2	0.00	0.00	0.00
12	Default	Home Owner	Marital Status	Income	Default						
13	No		Married	Average	Yes	1	0	1	0.00	0.00	0.00
14	Default	Home Owner	Marital Status	Income	Default						
15	No		Divorced	Average	Yes	0	1	1	0.00	0.00	0.00
16											

Source. Microsoft product screen shot reprinted with permission from Microsoft Corporation.

In the third class period, we discussed a case that we had prepared based on "A Well-Known Business School" case by Bell (1998). The original intent of this case is to create expert systems. In our version of the case, the focus is on data mining and classification, and the task is to extract rules for admitting students into an MBA program from a database of previous MBA applicants. The database contains records for 73 previous MBA applicants and each record contains an applicant's GPA and GMAT scores, the number of months of relevant work experience, an extracurricular activity score, an essay score, and whether or not the applicant was offered admission into the program. The GPA and GMAT scores are continuous variables. The work experience variable has integer values ranging from 0 to 84 months. The activity and essay variables are rated as A, B, C, or D with A being the highest score and D the lowest.

This database has two attributes that are continuous variables and several records with missing values. The continuous variables and missing values issues provide an excellent opportunity to discuss data preparation, which is one of the KDD processes. (Bell 2008, p. 30) refers to this data preparation step as "pre-O.R.," which he defines as "the grunt work that O.R. people have to do before they can apply O.R. methods and models." Based on his experience, Bell (2008) estimates that there is an "80/20 rule of analytics": Analysts usually spend a lot more time doing data processing than building the actual OR/MS models. As such, Bell believes that students should be given more exposure to this exercise in their O.R. coursework and that this MBA database provides an excellent opportunity to do so.

We began the data preparation process by instructing students to check the validity of their data. For example, GPA scores should be between 1 and 4. This can be verified easily using the MIN and MAX functions in Excel. We also showed the students how to perform this data verification process using the sorting tool and the AutoFilter tool in Excel.

While performing the data validation process, many students noticed that there were records with missing or incomplete GPA values. When we asked them to suggest ways to deal with these records, a common suggestion was to remove these records from the database. We responded by posing a question based on the bank loan problem they had previously seen: If a customer were to refuse to provide information about his income in the loan application form, what could that imply about the customer? The students inevitably realized that a missing value could itself prove to be useful information. We then discussed possible solutions to handling missing values, for example, treating a missing value as a valid stand-alone class value for the attribute or replacing the missing value with an appropriate estimate such as the average, median, or mode value, which can be based on the whole database or a sample of records with similar values for the other attributes.

Students usually run into a minor roadblock when verifying the GMAT variable because they are unaware of the range for GMAT scores. This leads to another important lesson: Understand your data. From a quick Web search, students found that GMAT scores ranged from 200 to 800. Recalling that decision trees work better with categorical instead of continuous variables, students were asked to suggest rules to discretize the GMAT variable. Without fail, they suggested using "nice" ranges such as 200 to 300, 300 to 400, and so on. Even though such a discretization rule seems reasonable, a more important factor is whether the rule is sensible. When we explained that GMAT scores are interpreted in a similar fashion to ACT and SAT scores with which they are more familiar, they realized that each GMAT score corresponds to a percentile score and that a more sensible approach is to discretize the GMAT variable based on percentiles instead of the raw scores. Again, this reinforced the need to understand the data. At this point, the students are reminded about how the discretization decision could affect the final decision tree in terms of its size: broad versus narrow decision trees (based on the number of class values for each attribute) and shallow versus deep decision trees (based on the number of attributes).

With the necessary foundation and background from the discussion of the case and data, the students gathered in their groups to decide how to preprocess their data and build their decision trees. While working on their decision trees, the groups found that a handful of the nodes at the end of their decision trees were not pure and they did not have any other attributes that they could use to split the impure child nodes. We discussed some ways to deal with these nodes, for example, implementing a "majority rules" or "flip a coin" rule to classify all the records in an impure child node, or using an approach called pruning that helps simplify the final decision trees so that more interpretable classification rules can be obtained.

On the day of the group presentations, we invited the MBA director to attend the class to provide real-life feedback and comments about the students' work and participate in the question-and-answer period. The students particularly enjoyed discussing the admission process with the MBA director. One interesting question asked by a student was how the director decided which applicants to admit if the number of qualified candidates exceeded the number of available spots in the program. From a data mining perspective, we mentioned that many classification algorithms including decision trees can also rank the records within the database (Ataman et al. 2006, Caruana et al. 1996, Crammer and Singer 2002, Rakotomamonjy 2004). Ranking using decision tree algorithms is typically done by modifying the algorithm so that in addition to providing the predicted class label for a record, the tree provides the probability of the record belonging to a class. These probabilities then can be used to order the data points (Provost and Domingos 2003).

As part of the summary of the data mining topic, the student groups were provided with a test or out-of-sample set of 20 applicants. The groups then assessed which of these 20 applicants would be accepted into the MBA program based on their decision trees. After the groups had classified the applicants in the test set, they were informed which applicants were accepted and which were not. Based on these results, we evaluated the accuracy of their decision tree models, where accuracy is defined as the ratio of correctly classified records to the total number of records:

$$\frac{TP + TN}{TP + FP + TN + FN},$$

and where TP, TN, FP, and FN are the number of true positive, true negative, false positive, and false negative records, respectively. These four metrics also allowed us to revisit the Types I and II error measures that the students have seen in their statistics courses.

We ended the lesson by instructing the student groups to construct a confusion matrix (Kohavi and Provost 1998) that displays the four metrics TP, TN, FP, and FN in a 2-by-2 table as shown in Figure 7.

Students recognize that the confusion matrix is similar to the table of joint probabilities that they had seen in sequential decision making problems, which is covered in the decision theory portion of the course. Recall that in sequential decision making, a decision maker is faced with choosing from two or more competing options, where each option will result in different payoffs or rewards depending on the realization of the random event following the choice. The decision-making process may also include an option where the decision maker can enlist the help of an external source (for example, an expert) to provide better information about the likelihood of the occurrences of future random events. Typically, this information is presented in the form of conditional probabilities or in the form of joint probabilities similar to those in the confusion matrix table. This classification exercise illustrated to the students one approach of obtaining this expert information.

Figure 7 **Confusion Matrix**

		Predicted	
		Negative	Positive
Actual	Negative	TN	FP
	Positive	FN	TP

6. Conclusions

In this paper, we introduce a classification algorithm called decision tree induction that can be used for data mining. We show how one can implement the algorithm in Microsoft Excel and discuss our experiences teaching this material within an undergraduate management science elective course.

Data mining is usually not covered in the typical OR/MS curriculum. However, we believe that data mining is a very useful and practical tool that students should have in their OR toolbox and, therefore, it is worth dedicating a handful of class hours to this increasingly important topic. We envision that this material can be a supplemental topic to forecasting (e.g., classification as a predictive task), regression (e.g., an alternative approach to binary logistic regression for identifying the key independent variables to help explain the dependent variable), or decision theory (e.g., creating the confusion matrix for use in sequential decision-making problems).

Acknowledgments

We thank the editor and anonymous referees for their thoughtful comments on earlier versions of the paper. Their suggestions helped improve the content and presentation of the paper.

References

Albright, S. C., W. Winston, C. Zappe. 2008. *Data Analysis and Decision Making with Microsoft Excel*. South-Western College Publishers, Cincinnati, OH.

Anderson, D. R., D. J. Sweeney, T. A. Williams, J. D. Camm, R. K. Martin. 2010. *An Introduction to Management Science*. South-Western College Publishers, Cincinnati, OH.

Ataman, K., W. N. Street, Y. Zhang. 2006. Learning to rank by maximizing AUC with linear programming. *Proc. IEEE Internat. Joint Conf. Neural Networks* (*IJCNN*), Vancouver, British Columbia, Canada, 123–129.

Babcock, C. 2006. Data, data, everywhere. *InformationWeek*. Accessed July 6, 2010, http://www.informationweek.com/news/global-cio/showArticle.jhtml?articleID=175801775.

Bell, P. C. 1998. *Management Science/Operations Research: A Strategic Perspective*. South-Western College Publishers, Cincinnati, OH.

Bell, P. C. 2008. Riding the analytics wave. *OR/MS Today* **35**(4) 28–32.

Bershady, M. A., A. Jangren, C. J. Conselice. 2000. Structural and photometric classification of galaxies. I. Calibration based on a nearby galaxy sample. *Astronomical J.* **119** 2645–2663.

Caruana, R., S. Baluja, T. Mitchell. 1996. Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. *Adv. Neural Inform. Processing Systems* **8** 959–965.

Crammer, K., Y. Singer. 2002. Pranking with ranking. *Adv. Neural Inform. Processing Systems* **14** 641–647.

Dasarathy, B. V. 1990. *Nearest Neighbor* (*NN*) *Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA.

Domingos, P., M. Pazzani. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learn.* **29**(2–3) 103–130.

Fayyad, U., G. Piatetsky-Shapiro, P. Smyth. 1996. From data mining to knowledge discovery in databases. *AI Magazine* **17**(3) 37–54.

Hillier, F. S., G. J. Lieberman. 2009. *Introduction to Operations Research*. McGraw-Hill, New York.

Kohavi, R., F. Provost. 1998. Glossary of terms. *Machine Learn.* **30** 271–274.

MacKay, D. J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK.

Mangasarian, O. L., W. N. Street, W. H. Wolberg. 1995. Breast cancer diagnosis and prognosis via linear programming. *Oper. Res.* **43**(4) 570–577.

Netflix. 2006. Netflix prize. Accessed August 18, 2009, http://www.netflixprize.com.

Olson, D., Y. Shi. 2007. *Introduction to Business Data Mining*. McGraw-Hill, New York.

Pantel, P., D. Lin. 1998. Spamcop: A spam classification and organization program. *Proc. Fifteenth Natl. Conf. Artificial Intelligence, Madison, WI*, 95–98.

Provost, F., P. Domingos. 2003. Tree induction for probability-based ranking. *Machine Learn.* **52** 199–215.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learn.* **1** 81–106.

Raffensperger, J. F., R. Pascal. 2005. Implementing dynamic programs in spreadsheets. *INFORMS Trans. Ed.* **5**(2). http://ite.pubs.informs.org/Vol5No2/RaffenspergerRichard/.

Ragsdale, C. 2007. *Spreadsheet Modeling & Decision Analysis: A Practical Introduction to Management Science*. South-Western College Publishers, Cincinnati, OH.

Rakotomamonjy, A. 2004. Optimizing area under ROC curve with SVMs. *First Workshop ROC Analysis in AI, Valencia, Spain*, 71–80.

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.* **65** 386–408.

Rumelhart, D. E., G. E. Hinton, R. J. Williams. 1986. Learning internal representations by error propagation. D. E. Rumelhart, J. L. McClelland, eds. *Parallel Distributed Processing*, Vol. 1. MIT Press, Cambridge, MA, 318–362.

Tan, P.-N., M. Steinbach, V. Kumar. 2006. *Introduction to Data Mining*. Addison Wesley, Boston.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Wei, C., I. Chiu. 2002. Turning telecommunications call details to churn prediction: A data mining approach. *Expert Systems Appl.* **23** 103–112.

Witten, I. H., E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, San Francisco.