# Detecting Symptoms of Low Performance Using Production Rules

Javier Bravo and Alvaro Ortigosa
{javier.bravo, alvaro.ortigosa}@uam.es
Computer Science Department, Universidad Autónoma de Madrid

**Abstract.** E-Learning systems offer students innovative and attractive ways of learning through augmentation or substitution of traditional lectures and exercises with online learning material. Such material can be accessed at any time from anywhere using different devices, and can be personalized according to the individual student's needs, goals and knowledge. However, authoring and evaluation of this material remains a complex a task. While many researchers focus on the authoring support, not much has been done to facilitate the evaluation of e-Learning applications, which requires processing of the vast quantity of data generated by students. We address this problem by proposing an approach for detecting potential symptoms of low performance in e-Learning courses. It supports two main steps: generating the production rules of C4.5 algorithm and filtering the most representative rules, which could indicate low performance of students. In addition, the approach has been evaluated on the log files of student activity with two versions of a Web-based quiz system.

## 1. Introduction

Modern information technologies have found their ways into the classrooms: new applications (learning management systems, virtual labs, etc.), new devices (PDAs, Smartphones), new protocols (SMS, Bluetooth) are used by teachers and students in real-life education [4]. These technologies have facilitated the wider adoption of online e-Learning systems in the last decade. Among other benefits of these systems, we can obtain more interactivity, reach learning experience, flexibility of access, etc. However, design and evaluation of e-Leaning systems yet remain complex tasks that require both time and expertise that many classroom teachers do not have. In most of cases instructors or course designers need to design his/her e-Learning course. Development of better technologies for authoring and evaluation of e-Learning systems is a very important research and practical problem, which becomes even more challenging, when the technology uses classroom instructors as its target audience.

In this paper we try to address this problem by proposing an approach to evaluation of e-Learning systems based on the analysis of log data. Evaluation of e-Learning systems is a complex and time-consuming task. One of the challenges for an instructor evaluating an e-Learning system is lack of evidence from students, since he/she has access only to their interactions with the learning material and cannot observe their behaviors, or receive the feedback from them in a timely manner. Due to the fact that student's behaviors are hidden inside their interactions, instructors should analyze them in order to assess the performance of students and evaluate the learning software. Furthermore, e-Learning systems generate vast quantity of data (log files). These data consist of records of students' actions within the system. Since the log files are often very big, traditional data

analysis tools and techniques are not always useful. Therefore, an alternative technology needs to be applied.

We considered that data mining methodology is more adequate for this task, because it is a technology that blends traditional data analysis methods with sophisticated algorithms for processing large volumes of data in order to discover meaningful information [11]. Our work is focused on evaluating an e-Learning system by exploring its usage logs in order to find patterns that could indicate low performance. It is worth to mentioning that these patterns are called *symptoms*. An example of a symptom is a significantly higher number of failures for a given exercise than the average number of failures for this exercise. It is difficult for instructors to detect symptoms of this kind without the help of data mining methods. At the same time, it can be highly beneficial for an instructor to be aware that a certain student has problems with particular set of exercises, and hence be able to intervene. How do such symptoms occur? What factors can trigger them? The present work tries to answer these questions by providing a method for detecting symptoms of low student performance in the course.

Naturally, not every symptom observed in the real data, is an indicator of variations in performance in the course. It is useful to identify the relevant symptoms causing drops in performance; it is also useful to distinguish the most important of them. For these purposes, we have developed a method helping to filter and rank relevant symptoms. The method is based on production rules of C4.5 algorithm.

This paper is organized as follows. The next section presents a brief overview of related work. Section 3 describes the data set we have used in this study. The proposed approach, as well as the results of the evaluation are detailed in Section 4. Finally, the last section concludes the paper with discussion and plans for future work.

## 2. State of the Art

Over the last years data mining has become a very popular technology in many areas of information sciences including e-Learning. Romero and Ventura provide a comprehensive overview of data mining applications for education [8]. Large pool of student activity has been collected by many research facilities. A good example is the open data repository *DataShop* that stores over 110 datasets containing nearly 18 million student actions [5].

Many researchers recognize the potential of data mining methods to help diagnose problems in e-Learning applications. For example, Ueno proposes a method for detecting irregular learning processes using student response time [12]. The method uses Bayesian predictive model and parametric statistical tests to identify potential outliers. Our approach it is somewhat similar to Ueno's work; however, it is based on the use of decision trees and the analysis of students' answers to learning problems. While Ueno's system aims to help students, the goal of our project is to help course designers to design and evaluate e-Learning courses.

Another relevant work has been done by Meceron and Yacef [6]. They propose to use the *cosine* and *lift* as two alternative measures of interestingness for association rules instead of *confidence* and *support*. These alternative measures are very successful in filtering uninteresting *association rules*. The selection of the interesting or relevant rules is one of the foci of our work, but in our case we filter production rules instead of association rules.

## 3. Data Description

The input data for this project have been provided by the School of Information Sciences at University of Pittsburgh (Pittsburgh, USA). The data were collected through several semesters of students' interaction with *QuizGuide* and *QuizPACK* systems within the framework of introductory programming course. Only the data in 2007 were selected for this work, since the students of different years are not comparable because the contents of the course are different by each year. Consequently, records corresponding to 55 students of seven different groups were selected. They took an adaptive course of "Introduction to C programming" generating **52734** interactions with the system.

### 3.1 QuizGuide and QuizPACK: providers of the data

*QuizGuide* is a Web-based service that provides personalized access to self-assessment quizzes for C programming language [9]. It does not serve the quizzes itself; instead it stays as a wrapper between the quiz provider and the student's browser and augments the quizzes with adaptive navigation cues. To guide students to the right learning content QuizGuide exploits adaptive link annotation technique. Quizzes and questions in QuizGuide are annotated with adaptive icons. Every icon delivers to a student two kinds of information: his/her individual progress with the corresponding content, the relevance of the content to the student's current learning goal. The goal information is calculated based on the course schedule and the relation of the quizzes to different topics in the course. The progress information is calculated based on the individual student's history of correct and incorrect attempts for the corresponding question, as well as other questions covering the same domain concepts.

The quizzes are generated and evaluated by *QuizPACK* system [10]. Every QuizPACK question consists of a simple C program that students need to evaluate and answer what will be the output of the program or what will be the final value of the target variable. The important feature of QuizPACK is question generation. When the question is delivered to a student, one of the numeric values in the question text is dynamically instantiated with a random value. Consequently the students can try the same question again and again with different correct answers.

### 3.2 Log Data Description

When a student answers a question in QuizGuide a new log entry is added to the database. These entries consist of several fields. For our analysis we augmented the log entries with information about domain concepts. Table 1 shows three instances of the enhanced data log we analyzed (52734 cases).

***Table 1. Log entries of students' interactions with QuizGuide***

| userId | groupId | result | activity | concept | conceptParent | session | success |
|---|---|---|---|---|---|---|---|
| *uid_199* | *gid_190* | *0.00* | *2dimensional_array1* | *printf* | *output* | *21BC5* | *no* |
| *uid_199* | *gid_190* | *1.00* | *2dimensional_array1* | *printf* | *output* | *21BC5* | *yes* |
| *uid_359* | *gid_72* | *1.00* | *2dimensional_array1* | *printf* | *output* | *A0B33* | *yes* |

The header of Table 1 contains the following attributes:

- **userId**: id of the student in the system.
- **groupId**: id of the group to which the student belongs. Six groups belong to different colleges and one group to "world group". This last group is created for the students who did not belong to these colleges and follow the e-Learning course by free access.
- **result**: outcome of the student's attempt. Two values are possible: 0 (incorrect answer) and 1 (correct answer).
- **activity**: name of the quiz or activity.
- **concept**: name of the concept covered by the activity.
- **conceptParent**: name of the concept parent.
- **session**: id of the web session (a session combines students activity between login and logout).
- **success**: verbal representation of the field "result" (result = 1 corresponds to success = "yes"; result = 0 corresponds to success = "no").
- The system also stores the **time stamp** when the student started the activity.

It is important to point out that a student can perform the activity more than once. This situation can be observed in the first and second row of Table 1. For instance, the first row shows that the student *uid_199*, belongs to the group *gid_190*, obtained *0* (success = *no*) in the activity *2dimensional_array1*. This table also provides information between the activities and concepts, e.g., this activity is related to the concept *printf* and the general concept of this activity is *output*. In addition, this row provides information about the session id, *21BC5*. However, the second row exhibits that the same student solved the exercise successfully in another attempt. Thereby, interactions and cases are the same concepts, but students are not equivalent than cases since they can repeat the activities many times.

## 4. Analysis of the Data

The goal of this work is to analyze the log data and to find significant patterns of behavior, which can provide support for improving the teaching material. In this context, decision trees were selected as the data mining technique, since they produced good results in previous works [2]. In particular, the current work uses the **C4.5** algorithm [7].

The next step on the analysis process was to define which attributes from the log files would be analyzed and which one used as the class variable. Considering the intention of finding activities that can present difficulties for a subset of the students, two attributes were selected: *activity* (name of the activity) and *groupId* (group id). The idea was to detect activities that were significantly more difficult for students of a given group, compared to the students of the other groups. These attributes were considered enough for the task at hand, without adding excessive complexity to the problem. The class variable is *success* representing the outcome (is given as *yes* or *not* depending on whether the student solved the activity successfully or not) of students' attempts to answer QuizPACK question. Consequently, two classes compose the classification model, The space of the problem is 644* combinations, which together with the data size indicate that the data analysis is complex enough to apply the decision trees method.

The distribution of training data is shown in Table 2. The first row shows the values of the *success* attribute (class variable) and the number of cases covered by them. In this case, proportions of the classes *yes* and *no* are rather similar (46% versus 54%). This feature is suitable for building decision trees, which benefit from the balanced classes. The next seven rows provide the values of *groupId* and the number of cases for each of these values. It is clear that the group gid_67 has the lowest number of cases (70), and group gid_190 holds most of them (38382). Even though the data sets of groups with fewer cases could be removed, we decided to keep them, since every set of cases is valuable. Moreover, as it is more substantial for this research the model might be able to represent every set of data than provides better predictions. Finally, the last row of the table 2 offers the names of activities available (46 activities) in the adaptive course "Introduction to C programming". For example, the first three activities (*2dimensional_array1, 2dimensional_array2 and 2dimensional_array3*) range over exercises of coding arrays in C language.

***Table 2. Distribution of training data***

| Properties | Values | Number of cases |
|---|---|---|
| *success* | yes | 24010 |
| | no | 28724 |
| *groupId* | gid_67 | 70 |
| | gid_72 | 1629 |
| | gid_190 | 38382 |

* The space of the problem is defined by the whole combinations of the attributes. In this case, *activity* variable has 46 different values, *groupId* has seven values, and two classes are possible. As a result, the space of this problem is 46*7*2 = 644. It is worth to mentioning that the space is a measure of the problem complexity.

| | | |
|---|---|---|
| | gid_373 | 3879 |
| | gid_394 | 280 |
| | gid_441 | 6536 |
| | gid_442 | 1958 |
| *activity* | 2dimensional_array1,2dimensional_array2,2dimensional_array3, arithmetic_expression1,arithmetic_expression2, character_array1,character_array2,character_array3, character_processing1,character_processing2,character_processing3, conditional_operator1,complex_conditional1,complex_conditional2, function1,function2, function3,printing1,printing2, variable1,variable2,variable3,constant1,constant2, globvar_simplefunc1,globvar_simplefunc2, increment_decrement1,compound_assignment1, logical_expression1,logical_operator1,if_else1,if_else2, do1,do2,for1,for2,while1,while2, nested_loop1, switch1 pointer1,pointer2,pointer3,array1,array2,array3 | |

The third step in the analysis was to use the algorithm **C4.5** in order to obtain the decision tree. The pruning in the tree was deactivated, that is to say, the confidence factor (CF) is set to 100%. The reason of setting this CF is because overfitting does not represent a problem, since the resulting decision tree is not used to make predictions. Actually, the better model fits the training data, the better these data can be described by the model, which is the goal of this work. Another relevant parameter of the **C4.5** algorithm consists of grouping attribute values. This option supported two concerns: insufficiency of data and information-gain ratio criterion. The first concern is that useful patterns of the data are not detected due to insufficiency of data; therefore grouping values can get enough data for detecting these patterns. The second concern is related to the performance of information-gain ratio is lower when the attribute has many values [7, Chapter 5]. This option produces better results when the data contain discrete attributes with many values. In the case of this study, the attribute *groupId* contains 7 different values and the attribute *activity* has 47 values. As there are discrete attributes with many values in the data, this work exploits grouping attribute values option. As a result, applying the algorithm **C4.5** a decision tree with size of 168 is obtained. It is worth to bring out that the tree is composed by 113 leaves (54 leaves with class *no* and 59 with class *yes*) and 55 decision nodes.

Subsequently, the decision tree was analyzed. As the objective of this research is to find difficulties leaves with value *no* are more interesting than others. The initial attempt was to use the **key node** method [1], but this method requires analyzing every path from the leaf to the top of the tree in order to find the relevant decision nodes. Therefore, analyzing 54 paths requires to analyze each decision node of each path. One of the main problems is that some paths could be irrelevant or redundant, and **key node** method does not ensure avoid redundant paths. For this reason, the next attempt was to utilize an alternative method the *production rules*[Ψ] of **C4.5**. These rules are based on two

[Ψ] Quinlan defines a production rule as *left part* --> *right part* [7, p. 8-12]. The left-hand side contains the conditions and the right-hand side is referred to the class. If the case satisfies all the conditions (conjunction of attribute-based tests), it is classified by the value on the right part. For example, the rule *Rule 32: activity = variable2 --> class no [69.5%]* indicates if a case contains the activity *variable2* this case will be classified to class *no* with almost 70% of accuracy.

assumptions: the rules are easier to understand and the rules avoid redundant data. Every node in the decision tree has a context established by the tests' outcomes of the previous node. In this sense, a rule is easier to understand since the context is on the left side [7, Chapter 7]. The other assumption is related to the fact that the trees sometimes produce similar sub-trees. In that regard, the process of building the rules removes irrelevant conditions and avoids redundant rules. Applying the *C4.5rules* program to the data generated 20 rules. As in the case of decision trees only the rules in which the right side is *no* are selected (11 selected rules).

The last step was to filter the rules in order to select the relevant rules. Following this idea two approaches are possible: from the point of used cases (i.e. the cases in which all the conditions of the rule are satisfied) and from the point of accuracy (i.e. the percentage of correctly classified cases). Taking into account data description, we have chosen the former approach. *C4.5rules* also provides a ranking of non redundant rules based on the error rate (table 3 contains this information). It is important to note that three redundant rules corresponding to *no* class are not including in this ranking. For this reason, Table 3 presents eight rules with *no* class (they are highlighted in the table). The column **Rule** shows the id rule, the next column indicates the number of conditions in the rule, the column **Error** is an estimation of number of cases classified incorrectly, the column **Used** provides the used cases, the column **Wrong** exhibits the number of cases classified incorrectly, and the last column indicates the value of the class.

***Table 3.Ranking of rules***

| Rule | Size | Error (%) | Used | Wrong | Class |
|---|---|---|---|---|---|
| 21 | 2 | 0.0 | 74 | 0 | yes |
| 20 | 2 | 19.2 | 159 | 30 | yes |
| 13 | 2 | 22.5 | 60 | 13 | yes |
| 9 | 2 | 25.6 | 236 | 60 | yes |
| 25 | 1 | 32.5 | 3657 | 1189 | yes |
| 16 | 2 | 35.5 | 280 | 99 | yes |
| 40 | 1 | 38.9 | 16061 | 6526 | yes |
| 17 | 2 | 46.4 | 738 | 342 | yes |
| 3 | 2 | 47.3 | 278 | 131 | yes |
| 19 | 2 | 17.6 | **71** | 12 | **no** |
| 23 | 2 | 18.2 | **2608** | 475 | **no** |
| 8 | 2 | 20.9 | **242** | 50 | **no** |
| 33 | 1 | 29.8 | **1771** | 494 | **no** |
| 7 | 2 | 32.1 | **344** | 110 | **no** |
| 6 | 2 | 32.4 | **2508** | 813 | **no** |
| 12 | 2 | 34.2 | **20419** | 7522 | **no** |
| 14 | 2 | 39.7 | **253** | 100 | **no** |

As the filter criterion is based on used cases, the selected rules were ordered by this column. Another selection is needed because some rules are not enough representatives. This new sub-set of rules is obtained by using a lower band of used cases. Therefore, a rule is not enough representative if it is below this limit. Experiments with the data disclosed that the adequate lower band is calculated as 10% of sum of used cases of the rules. In this case, the sum of used cases is equal to 26445; thereby the lower band is

2645. Only three rules satisfied this threshold (rules 6, 23 and 12), hence they were selected. Table 4 demonstrates these rules.

***Table 4.Representative rules***

| | | | |
|---|---|---|---|
| Rule 6: | groupid in {gid_190, gid_373};<br>activity in {character_array2, printing2, while2} | -> | class no [67.6%] |
| Rule 23: | groupid in {gid_190, gid_441, gid_373, gid_394};<br>activity in {2dimensional_array2, do2, array1, array3} | -> | class no [81.8%] |
| Rule 12: | groupid = gid_190;<br>activity in {2dimensional_array1, 2dimensional_array2, 2dimensional_array3, arithmetic_expression1, arithmetic_expression2, character_array1, character_array2, character_array3, character_processing1, character_processing2, character_processing3, conditional_operator1, do2, function1, logical_expression1, nested_loop1, pointer2, pointer3, printing2, variable2, array1, array2, array3, do1, for1, for2, function3, if_else1, if_else2, logical_operator1, pointer1, while2, globvar_simplefunc2} | -> | class no [65.8%] |

Rule 6 indicates that most of the students from the groups 190 and 373 showed difficulties in the activities character_array2, printing2 and while2. This rule might indicate a symptom of low performance on these quizzes for these groups of students. The next rule demonstrates problems that the students from the groups 190, 441, 373 and 394 had when they where trying activities 2dimensional_array2, do2, array1 and array3. Hence, further attention should be paid to these groups and activities. Several questions can be asked. Are the exercises adequate? Were students presented with the necessary knowledge? Lastly, the rule 12 shows that students from group 190 experienced problems in the most of activities. This rule could indicate that the students in the group 190 did not have enough background knowledge to take the course "Introduction to C programming".

The previous analysis showed the whole process of achieving the representative symptoms, i.e. selected production rules. Therefore, the **proposed method** is summarized in the following lines:

- *Generate the production rules by using C4.5 algorithm.*
- *Select the rules of the ranking table in which the right part indicates a "failure" in activities. In our case, a failure is indicated by value no of the class success.*
- *Filter limit* $= \sum_{i=first-selected-rule}^{last-selected-rule} used - cases_i \Big/ 10$
- *Select the rules which cover a number of used cases greater than the filter limit.*
    - *After this selection is possible to obtain only one rule. In this case, it is advisable to select also the next closer rule to the filter limit.*

It is important to note that the information displayed in these last rules could be useful to instructors or course designers for enhancing their e-Learning courses. Thus, it would be

a valuable help to show this information to them, since they could evaluate their courses better by finding the last symptoms.

## 5. Conclusions and Future Work

This work has presented full analysis of real data of an e-Learning course. In this particular case 52734 instances were analyzed by using *decision trees* and *production rules*. The objective of the analysis was to find symptoms that could indicate presence of low performance in courses provided by *QuizGuide* and *QuizPACK* systems, i.e., to find particular activities in which a given category of students showed difficulties providing feedback to the instructors.

The analysis demonstrated that three rules indicate the presence of symptoms of drops in performance for several profiles of students. In fact, one of the rules exhibits that the group 190 showed difficulties in the course, since they failed the majority of the activities in the course. This information could be relevant for the instructor or designer of the e-Learning course, because he/she can improve it by adding new activities, and/or modifying existing activities or course structure. Furthermore, the other two rules showed that students who belong to groups 373, 441 and 394 presented problems in solving particular activities. This fact also indicates drops in the performance of the system for these groups of students. This information is also useful for the instructor or course designer, since he/she can modify these activities in order to improve the learning process for these particular students.

It is worth noting that this analysis can be performed with other type of data, even for data of different contexts. However, experiments with other types of data showed that the filter limit of this work is less accurate than others like third quartile of used cases.

Decision trees and production rules present some weaknesses. They are strongly related to the distribution of the data and proportion of classes. Therefore, small variations in the data could cause different conclusions. Consequently, the future work includes supporting these techniques with other Data Mining methods such as *association rules*. Other future line includes supporting the method described in this work in **ASquare** [3]. The goal of this tool is to provide a high abstraction level interface. Thereby, **ASquare** supplies more intelligible results to human beings; that is to say, a person without knowledge of Data Mining can understand the results.

Finally, it is important to test and validate the proposed analysis with other types of data. In this sense, future work also includes to apply this analysis for data of different educational systems in order to improve the detection method.

### Acknowledgements

This work has been funded by Spanish Ministry of Science and Education through the HADA project TIN2007-64718. We would like to thank the University of Pittsburgh for providing us the interaction data of their students. The authors would also like to thank to Leila Shafti who contributed in this work with her helpful comments.

## References

[1] Bravo, J., Ortigosa, A., and Vialardi, C. A Problem-Oriented Method for Supporting AEH Authors through Data Mining. *Proceedings of International Workshop on Applying Data Mining in e-Learning (ADML'07) held at the Second European Conference on Technology Enhanced Learning (EC-TEL 2007), 2007,* p. 53-62.

[2] Bravo, J., Vialardi, C., and Ortigosa, A. Using Decision Trees for Discovering Problems on Adaptive Courses. *Proceedings of E-Learn 2008: World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education*, 2008, p. 268-277.

[3] Bravo, J., Vialardi, C., and Ortigosa, A. ASquare: A Powerful Evaluation Tool for Adaptive Hypermedia Course System. *Proceedings of Hypertext Conference,* 2008, p. 219-220.

[4] Chen, F., Myers, B., and Yaron, D. Using Handheld Devices for Tests in Classes. *Tech. Rep. CMU-CS-00-152, Carnegie Mellon University School of Computer Science, and Tech. Rep. CMU-HCII-00-101, Human Computer Interaction Institute. Human Computer Interaction Institute*, 2000.

[5] Koedinger, K.R., Cunningham, K., Skogsholm, A., and Leber, B. An open repository and analysis tools for fine-grained, longitudinal learner data. *Proceedings of Educational Data Mining 2008: 1St International Conference on Educational Data Mining,* 2008, p. 157-166.

[6] Merceron, A. and Yacef, K. Interestingness Measures for Association Rules in Educational Data. *Proceedings of Educational Data Mining 2008: 1St International Conference on Educational Data Mining,* 2008, p. 57-66.

[7] Quinlan, J.R. C4.5: Programs for Machine Learning. *Morgan Kaufmann Publishers*, California, USA, 1993.

[8] Romero, C. and Ventura, S. Educational Data Mining: a Survey from 1995 to 2005. *Expert Systems with Applications*, 2007, 33(1), p. 135-146.

[9] Brusilovsky, P., Sosnovsky, S., and Shcherbinina, O. QuizGuide: Increasing the Educational Value of Individualized Self-Assessment Quizzes with Adaptive Navigation Support. In J. Nall and R. Robson (Eds.), *Proceedings of E-Learn,* 2004, p. 1806-1813.

[10] Brusilovsky, P. and Sosnovsky, S. Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK. *Journal on Educational Resources in Computing (JERIC)*, 2005, 5(3), p. 6.1-6.22.

[11] Tan, P., Steinbach, M., and Kumar, V. Introduction to Data Mining. *Pearson-Addison Wesley Publishers*, Boston, USA, 2006.

[12] Ueno, M. Online outlier detection of learners' irregular learning processes. In Romero, C. and Ventura, S. (Eds.) *Data Mining in E-Learning*, 2006, p. 261-278.