KDD-99 Classifier Learning Contest
LLSoft's Results Overview

Itzhak Levin
LLSoft
6 Ha-Marganit Street
Rishon-Le-Ziyyon, 75427, Israel
il@llsoft.com

ABSTRACT

Kernel Miner is a new data-mining tool based on building the optimal decision forest. The tool won second place in the KDD'99 Classifier Learning Contest, August 1999. We describe the Kernel Miner's approach and method used for solving the contest task. The received results are analyzed and explained.

Keywords

Data Mining competition, decision trees, optimal decision forest, classification, prediction.

1. KERNEL MINER

Kernel Miner is a data-mining tool for the description, classification and generalization of data, and for predicting the new cases. Kernel Miner is a fully automated tool that provides solutions to database users. Although Kernel Miner applies a system of sophisticated mathematical models and algorithms, it is extremely simple for users. The tool has been developed for Windows 95/98/NT and works with different databases such as dBase, MS Access, SQL Server, Oracle, etc. directly or through ODBC or OLEDB.

2. GENERAL MODEL AND ALGORITHM

Kernel Miner is based upon the global optimization model developed. This global model is then decomposed into a system of interrelated, intercoordinated and interconsistent models and criteria. As a result, Kernel Miner constructs the set of locally optimal decision trees (the decision forest) from which it selects the optimal subset of trees (the subforest) used for predicting the new cases.

The predictive modeling technique is based on sophisticated methods for reduction of individual prediction results received from individual classification trees. This enables us to calculate the value of the global optimization criterion for any subset of trees.

The global optimization criterion is to minimize a value of the multiple estimator including the total cost of misclassifications, and taking into account parameters of reliability and stability for prediction. Here the notion of stability is analogous to the corresponding concept applied in the mathematical programming theory. Taking into account the parameters of reliability and stability for prediction enables us to avoid the overfitting problem.

Each constructed decision tree "covers" all the examples of the training data set. It is optimal in the sense of minimizing the above criterion given the initial "good" partition. To find the set of such partitions, a special optimization task is solved on the set of two- and three-dimensional contingency tables, where the last dimension is the dependent variable. Note that, in particular, the "good" partition can be defined by the found function of two independent numeric variables, so that the discretization of values of this function separates the values (classes) of the dependent variable "well". The notion of a "good" (and the best) splitting is used at different stages of the algorithm and is based upon a proprietary measure. The number of classification trees to be built is equal to the number of found initial "good" partitions. Although each decision tree is built optimal, we call it locally optimal because it is generally not unique element of the final optimal decision subforest. In particular (in the simplest cases), the optimal subforest may consist of one tree.

3. TASK

The task for the KDD'99 Classifier Learning Contest was to create a predictive model capable of distinguishing between legitimate (normal) and illegitimate (called intrusions or attacks) connections in a computer network. The training dataset consisted of about 5,000,000 connection records, and the training 10% dataset contained 494,021 records among which there were 97,278 normal connection records (i.e. 19.69 %). Each record contained values of 41 independent variables (fields) which described the different features of the connection, and the value of the dependent variable labeled as either normal, or as an attack, with exactly one specific attack type. Each attack type belonged to one of the following 4 categories:

1) probe, i.e. surveillance and other probing;

2) DOS, i.e. denial of service;

3) U2R, i.e. use-to-root;

4) R2L, i.e. remote-to-local.

The competitors were asked to predict the value of the dependent variable (normal or one of the above attack categories) for each record of the test dataset consisting of 311,029 records. The specific character of the task consisted of the following:

1. The test data was not from the same probability distribution as the training data.

2. The test data included specific attack types not in the training data.

3. It was required to predict attack categories while in the training dataset the attack types were also given.

4. APPROACH AND METHOD USED

We omitted a preprocessing stage for a lack of the domain knowledge. Fortunately, this was not required since Kernel Miner is able to find a full solution to the task automatically. All we specified was the matrix of misclassification costs.

We searched for the required patterns on the available training 10% dataset consisting of 494,021 records, for the following reasons: 1) The hypothesis that the training 10% dataset had been *randomly* chosen from the entire training dataset, was beforehand verified, and it was corroborated. 2) Taking into consideration that the number of independent variables as well as the number of different values of the dependent variable in the training dataset were relatively small, the available 10% dataset contained the quite sufficient number of records in order to construct the stable patterns and to draw the conclusions. 3) It was not required to create numerous samples in order to imitate the different probability distributions for the test data, since the Kernel Miner's predictive modeling technique does not assume that a test data must be from the same probability distribution as the training data. 4) The patterns found on the training 10% dataset were clear, precise, and stable. This could not be by chance. Moreover, this was also corroborated by the very high accuracy of prediction on the whole 5-million training dataset. If at least one of the above-mentioned facts did not take place, we would supplement our analysis with an investigation of the entire training dataset and/or different samples.

Kernel Miner's modeling automatic process consisted of the following stages:

4.1 Coding of Values of Variables

Note that in Kernel Miner the unique discretization for each numeric attribute doesn't exist. Discretization is carried out at each step of constructing any tree in an optimal way. This is necessary to ensure maximum possibilities for the further optimization.

4.2 Constructing the Set of Initial "Good" Partitions

Note that "goodness" of the initial partition is defined by not only the "goodness" of this partition itself but, finally, by the "goodness" of the decision tree to be constructed. Hence, it is important not to miss any initial "good" partition. Moreover, for example, the discussed set for the "normal" category significantly differed from the sets of initial partitions for each of the attack categories. Therefore, Kernel Miner constructed a broad set of such partitions (and consequently, trees), and only at the final stage selected the optimal subset of them. This is similar to seed selection - the best tree does not always grow from the best of available seeds.

The specific character of the task required to build trees not only for the different categories but for the specific attack types as well. Consider why this was necessary. The general pattern for each category was necessary for prediction since the test data included specific attack types not in the training data. This is one but not a single reason. At the same time, the majority of the known attack types were described by very precise patterns. The existence and joint application of general and special patterns sharply increased a confidence level of the prediction results.

4.3 Constructing the Decision Trees

After the selected seeds (the conditions defining the initial "good" partitions) had been sowed, Kernel Miner constructed an optimal tree from each of them. (Note that a proprietary criterion for "goodness" of a tree is mutually consistent with the global optimization criterion.) Each constructed tree "explained" all the records of the training data set. The decision forest that was built by Kernel Miner, contained in total 218 decision trees for categories (including the normal one), and 537 decision trees for specific attack types.

For example, consider the attack type "smurf" belonging to DOS category.

In the training 10% dataset more than half of all records (280790, or 56.8%) were labeled "smurf". Here is only two simplest trees each of which in a one-to-one manner defines the smurf records (see Figure 1 and Figure 2).

Note that each of these trees can be reformulated as the necessary and sufficient condition for existence of smurf records. For example, the first tree says that:

the type is "smurf" if and only if (519 < **src_bytes** <= 1032) and (**service** is **ecr_i**) .

Of course, all was not so simple. The majority of trees constructed were of substantially more complex structure, and did not enable the program to draw such one-valued conclusions. The volume restriction does not allow us to consider solving the task in details here.

4.4 Selection of the Optimal Decision Subforest

Now, like a sculptor, Kernel Miner must cut off superfluous. To solve this problem, Kernel Miner applied a fast approximate algorithm developed for solving the zero-one integer programming model. As a result, the clear patterns were determined for each category and the majority of specific attack types. For instance, for the above example Kernel Miner determined the pattern including 5 decision trees each of which could be reformulated as the necessary and sufficient condition for smurf records. Why did Kernel Miner select five trees? Why not just choose one of them? The answer is that Kernel Miner takes into account parameter of reliability for prediction. And it was not mistaken. It is natural to assume that each necessary and sufficient condition must be fulfilled at the exactly same records of the test dataset. But this was not so. The first necessary and sufficient condition was *(continued after smurf figures…)*

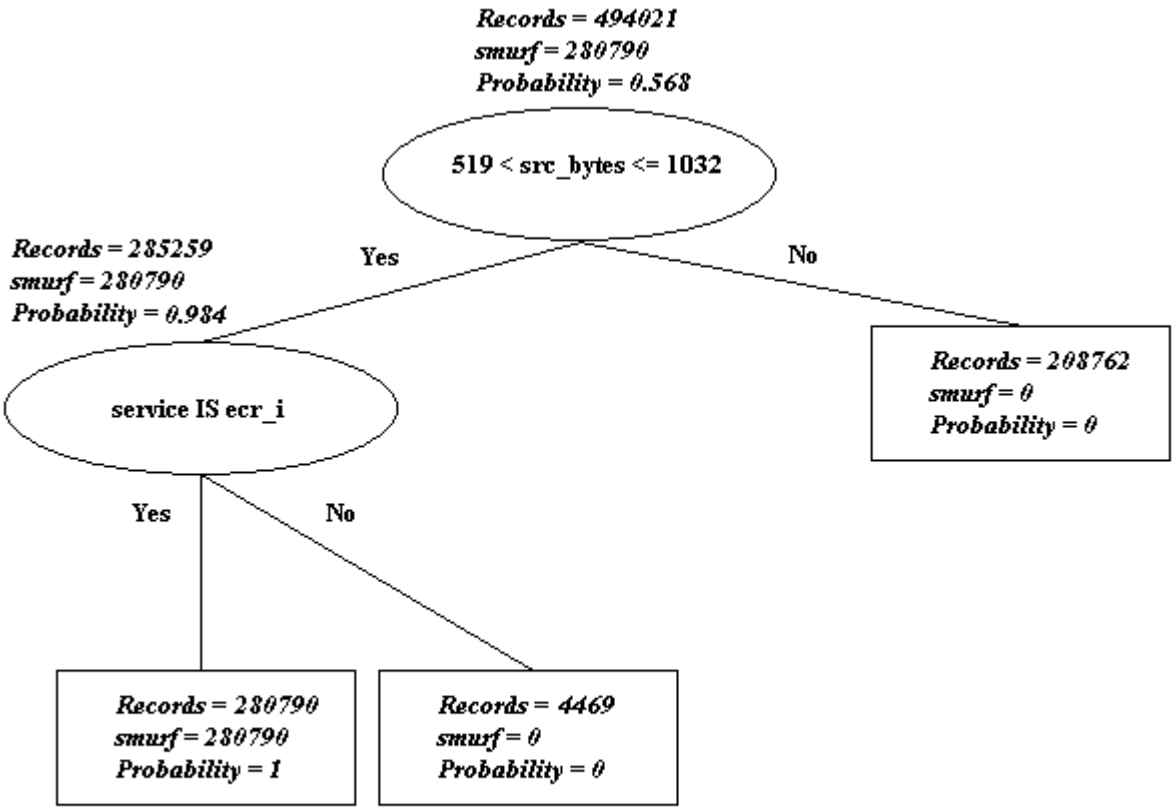

Figure 1: Decision Tree No. 1 for "smurf" pattern

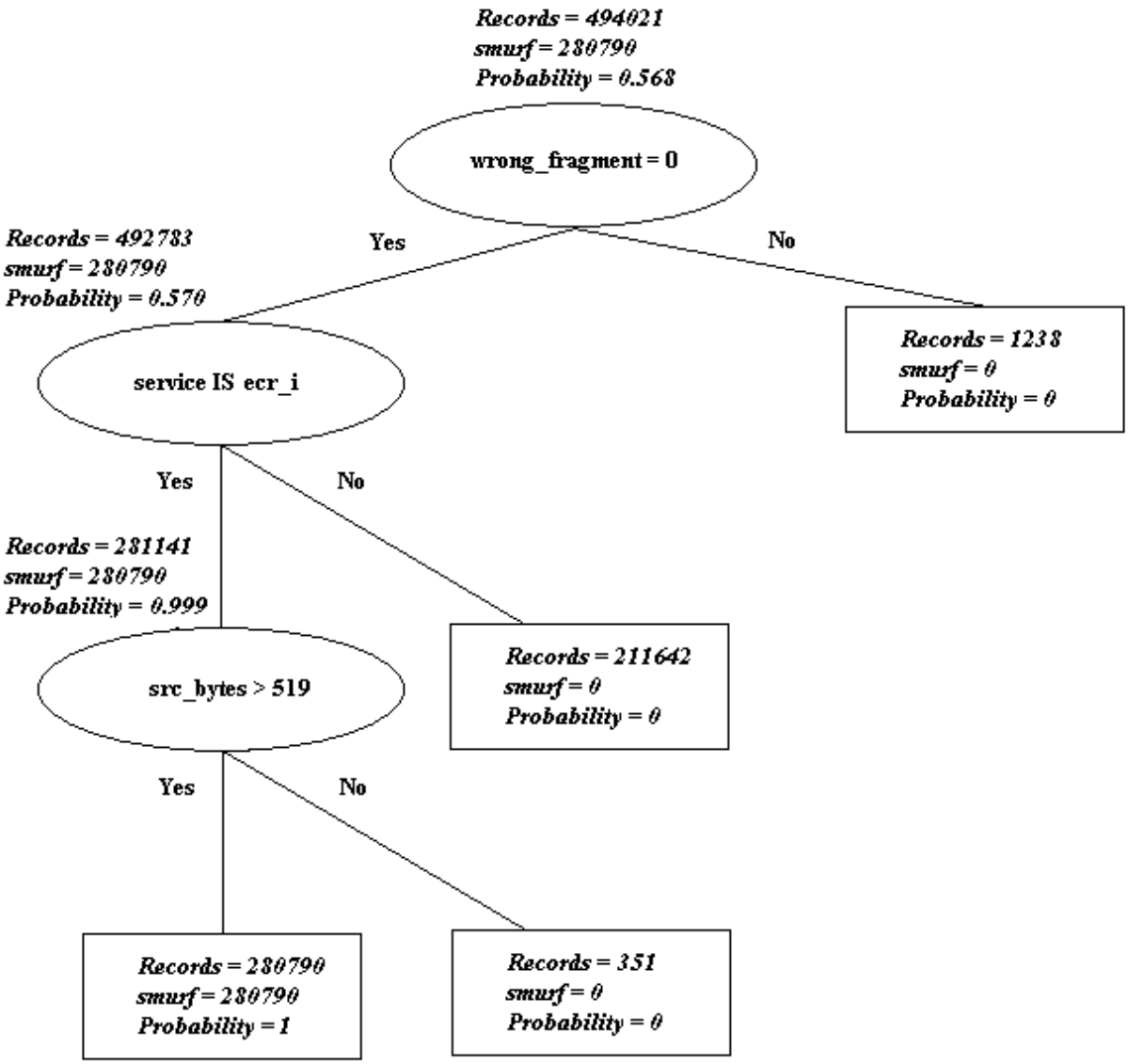

Figure 2: Decision Tree No. 2 for "smurf" pattern

fulfilled at 157,830 test records while the second one was fulfilled at 157,846 test records. Of course, we see extremely insignificant difference here, but we have considered this example for the illustration purpose solely.

4.5 Prediction on the Test Dataset

Based upon the found patterns, Kernel Miner applied its predictive modeling technique.

Kernel Miner single-valued determined the values (classes) of the dependent variable for 305,596 test records out of a total number of 311,029. Degree of confidence and parameter of reliability for these predicted test examples were very high. For the rest 5,433 test records, Kernel Miner applied the sophisticated methods for resolving conflicting cases.

The **hardware** used consisted of one PC (Pentium II 350 MHz) with 128 MB of RAM.

It took in sum about 22 hours of machine time to complete the whole process of finding all the patterns. Note that finding the patterns for specific attack types took more than 70% of this time. Prediction on the test dataset was very fast. It took about 20 minutes to read the data, and less than 5 minutes to recognize all the test examples.

5. ANALYSIS OF RESULTS

Our prediction results achieved an average cost of 0.2356 per test example and obtained the following confusion matrix:

(See Table 1)

The top-left element of the confusion matrix shows that 60,244 of the actual "normal" test examples were predicted to be normal. The last column indicates that in total 99.42 % of the actual "normal" examples were recognized correctly. The bottom row shows that 73.95 % of the test examples said to be "normal" were indeed "normal" in reality.

A sum of diagonal elements of the confusion matrix indicates the total number of the test examples that were predicted correctly. So, Kernel Miner correctly recognized 289,006 out of 311,029 test examples, i.e. 92.92 %. A comparison between our and the winning results shows that the total number of the test examples, recognized correctly by Kernel Miner, is greater than the corresponding number for the winning entry by 657 test examples (289,006 versus 288,349). At the same time, Kernel Miner made less misclassifications by 657 errors (22,023 versus 22,680). However, we made more misclassifications in element (R2L, Normal) of the confusion matrix (14,994 versus 14,527) which were evaluated by the highest cost (see the cost matrix Table 3). That is what caused the final result.

An analysis of our misclassifications shows that the majority of them belong to the new attack types which were not in the training data. There were in total 18,729 records belonging to such attack types in the test dataset. 16,815 out of them were recognized incorrectly by Kernel Miner, i.e. 89.8 %. More specifically, 16,387 records belonging to new attack types corresponded to the "normal" pattern with a high probability, but all of them were not indeed "normal" in reality. Perhaps, "something" existed that was not found by us. But in this case, to recognize the discussed test examples correctly, the new patterns and arguments must be "stronger" than the patterns found for "normal" records.

If to remove the records labeled the new attack types from the test dataset, then we get the following confusion matrix for Kernel Miner's results:

(See Table 2)

In this matrix we see the single large off-diagonal element, namely, (R2L, Normal). That is, we made many (4804) errors predicting "normal" for "R2L" records. Let us analyze this situation. The majority of these records were labeled "guess_passwd" in the test dataset (4110 out of 4804). Note that in the training 10 % dataset, there were only 53 records labeled "guess_passwd", i.e. only 0.01 % of all records. In spite of this, Kernel Miner determined the likely precise pattern for such records consisting of 10 decision trees. Each of these trees can be reformulated as the sufficient and "almost" necessary condition (although, of course, these trees are not independent). Here is two of these trees (See Figure 3 and Figure 4).

Consider now 4110 test examples labeled "guess_passwd", where we erroneously predicted "normal". All these records satisfied the condition that they are *not* "guess_passwd". All the attack categories said with confidence that these records do *not* belong to them. And simultaneously, these records had different high probabilities that they are normal. This is an explanation for our prediction. Apparently, here we came across the overfitting phenomenon - there were not enough "guess_passwd" records in the training dataset, in order to trust the pattern determined based upon them. In other words, the decision trees above overfit the data. It is also probable that the "guess_passwd" examples are so similar to "normal" connection records that the relevant formal mathematical patterns simply don't exist, and only the domain knowledge of intrusion experts and their intuition may help in this case.

(Continued after tables and guess_passwd figures…)

Predicted Actual	Normal	Probe	Dos	U2R	R2L	Total	% correct
Normal	60244	239	85	9	16	60593	99.42 %
Probe	458	3521	187	0	0	4166	84.52 %
Dos	5595	227	224029	2	0	229853	97.47 %
U2R	177	18	4	27	2	228	11.84 %
R2L	14994	4	0	6	1185	16189	7.32 %
Total	81468	4009	224305	44	1203	311029	
% correct	73.95 %	87.83 %	99.88 %	61.36 %	98.50 %		

Table 1: Confusion Matrix Obtained by Kernel Miner's Entry

Predicted Actual	Normal	Probe	Dos	U2R	R2L	Total	% correct
Normal	60244	239	85	9	16	60593	99.42 %
Probe	4	2370	3	0	0	2377	99.71 %
Dos	10	10	223278	0	0	223298	99.99 %
U2R	19	0	0	18	2	39	46.15 %
R2L	4804	3	0	4	1182	5993	19.72 %
Total	65081	2622	223366	31	1200	292300	
% correct	92.57 %	90.39 %	99.96 %	58.07 %	98.50 %		

Table 2: Confusion Matrix for Known Attack Types

Predicted Actual	Normal	Probe	Dos	U2R	R2L
Normal	0	1	2	2	2
Probe	1	0	2	2	2
Dos	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

Table 3: Cost Matrix

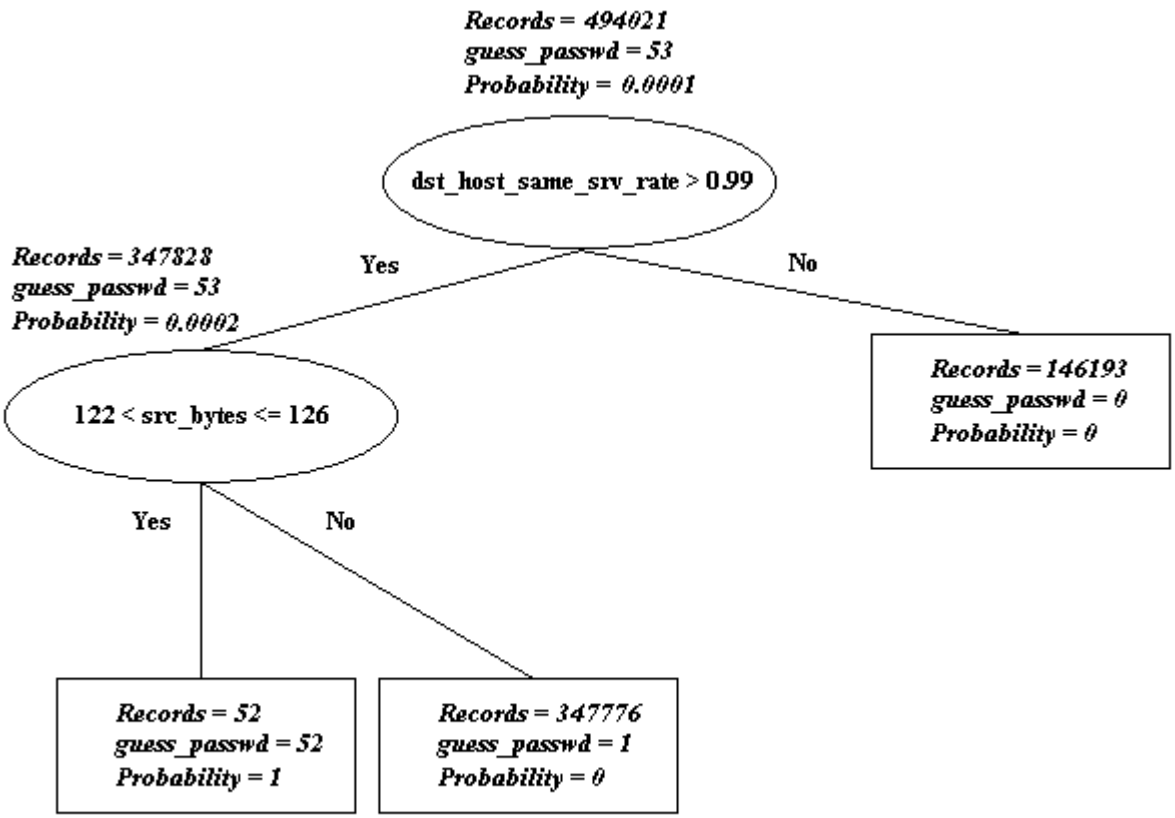

Figure 3: Decision Tree No. 1 for guess_passwd pattern

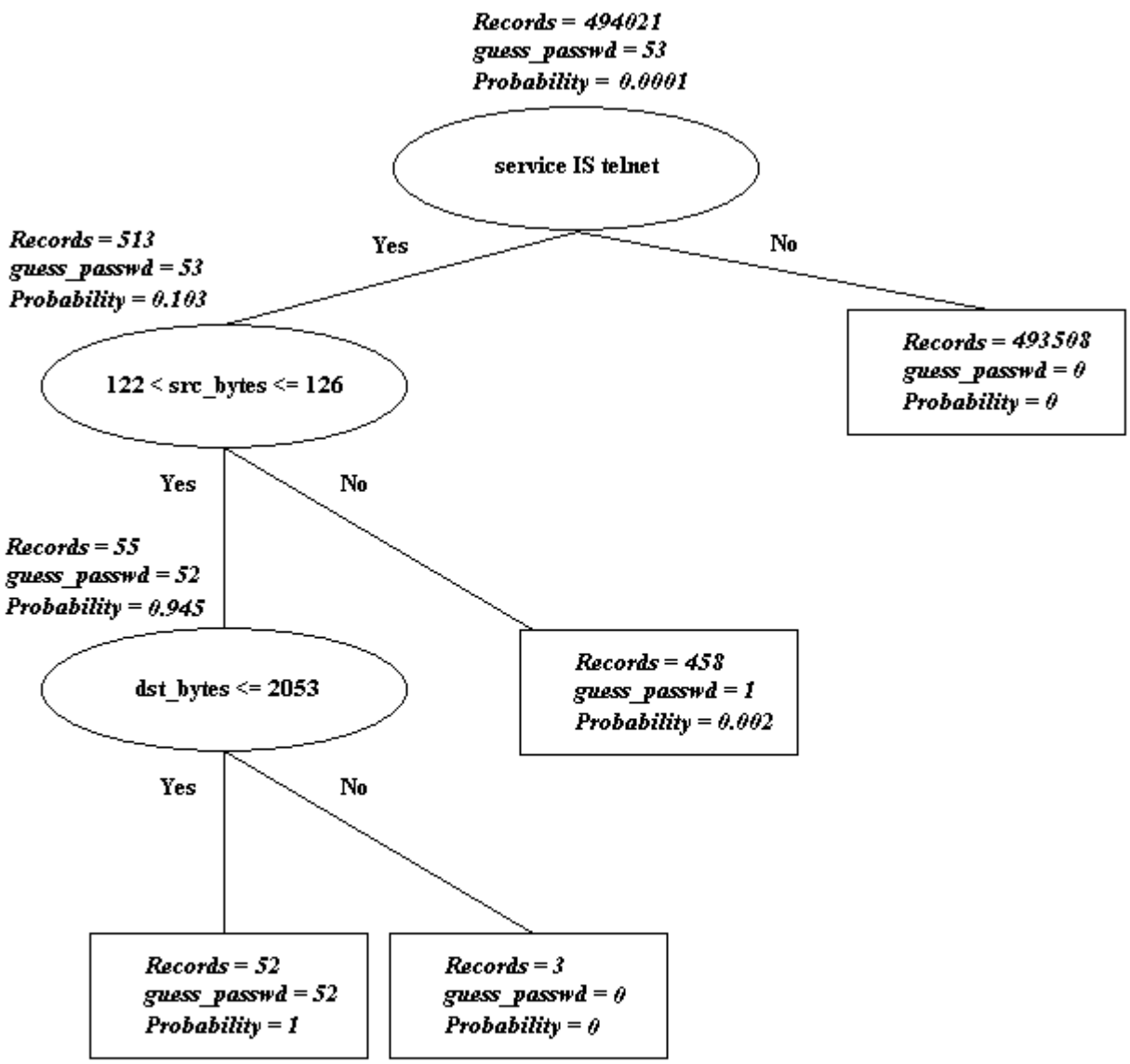

Figure 4: Decision Tree No. 2 for guess_passwd pattern

6. CONCLUSION

Kernel Miner is a new data-mining tool, and this was the first competition in which we participated. Our KDD CUP success has confirmed correctness and fundamental character of a series of new and known ideas that have been realized in our tool. The present version of the application is ready-to-use. At the same time, it should be noted that Kernel Miner is continually developing tool, and a series of our new additional methods and algorithms (including the parallel ones) are planned to be realized in the next versions of the tool.

About the author:

Itzhak Levin, M.Sc. in Applied Mathematics and Automized Information Systems, founded LLSoft Inc. in 1999. He is currently a CEO of LLSoft. The company specializes in the development of mathematical algorithms and software mainly for data mining and knowledge discovery area.