In **Nineth Panhellenic Conference in Informatics**, (Pitas, I. and Margaritis, K., Eds.), Thessaloniki, Greece, Greek Computer Society (EPY), November 2003, pp.346-360.

GenMiner: A data mining tool for protein analysis

Gerasimos Hatzidamianos[1], Sotiris Diplaris [1,2], Ioannis Athanasiadis [1,2], Pericles A. Mitkas [1,2]

[1] Department of Electrical and Computer Engineering

Aristotle University of Thessaloniki, 54124, GREECE

[2] Laboratory of Intelligent System and Software Engineering,

Informatics and Telematics Institute/ CERTH, 57001Thessaloniki, Greece.

Abstract. We present an integrated tool for preprocessing and analysis of genetic data through data mining. Our goal is the prediction of the functional behavior of proteins, a critical problem in functional genomics. During the last years, many programming approaches have been developed for the identification of short amino-acid chains, which are included in families of related proteins. These chains are called motifs and they are widely used for the prediction of the protein's behavior, since the latter is dependent on them. The idea to use data mining techniques stems from the sheer size of the problem. Since every protein consists of a specific number of motifs, some stronger than others, the identification of the properties of a protein requires the examination of immeasurable combinations. The presence or absence of stronger motifs affects the way in which a protein reacts. GenMiner is a preprocessing software tool that can receive data from three major protein databases and transform them in a form suitable for input to the WEKA data mining suite. A decision tree model was created using the derived training set and an efficiency test was conducted. Finally, the model was applied to unknown proteins. Our experiments have shown that the use of the decision tree model for mining protein data is an efficient and easy-to-implement solution, since it possesses a high degree of parameterization and therefore, it can be used in a plethora of cases.

Keywords: Databases and Information Retrieval, Bioinformatics, Data mining

Designated track: Case study

1. Introduction

A very important issue in bioinformatics is data mining in biological databases. Large databases were created for the recording and exploitation of biological data, due to the human DNA and protein decoding. In fact, over 20 trillion basic structural DNA elements are recorded in the three large biological databases (Genbank, EMBL, DDBJ) at that moment. Another crucial issue in bioinformatics is structural biology, that is the presentation of the structure of several biological macromolecules. The knowledge of the 3D structure of the macromolecules can give the answer to many diseases, since most of them are caused by malfunctions of the proteins that are related to them. The function of a protein is directly related to its structure. Bioinformatics cover the visualization of the 3D structure that has been derived from experimental data, as well as its prediction, using algorithms assumed to be valid for the protein structures. A protein is structured by amino-acids, whose sequence defines the protein's function. The quantity of amino-acids that participate in the structure of a protein varies and is dependent on its type. From the above, we can easily conclude the great importance that the proteins have for human. Until now the biological effect of proteins could be identified only through a time-consuming procedure and expensive experiments. Using data mining, this problem can be solved by implementing methods such as the one presented here.

Protein motifs are very critical for the prediction of a protein's function. At this time, a few databases containing motifs have been developed, such as Prosite, Pfam and Prints. Protein

functions combined with motifs are stored in these databases, thus allowing the knowledge of the probable function of the proteins containing certain motifs. A protein's function although can not be determined in this way, since the properties that characterize a protein is a function of many motifs where some overpower others. Thus, there is a vital need to create tools that can discover similarities in chains, consequently predicting a protein's behavior. In fact, data mining algorithms are successfully applied in unknown protein identification [1].
GenMiner integrates the three different stages required in order to apply data mining techniques in protein data. The first step in this procedure is data preprocessing. GenMiner is capable of processing the three major protein databases and export a file containing the information needed by various data mining algorithms[2]. The next step is the application of data mining techniques, provided by WEKA (Waikato Environment for Knowledge Analysis) [3] and MS SQL Analysis Manager 2000 [4]. GenMiner integrates these methods in a single Graphical User Interface, thus providing a robust data mining tool. As a final step, experiments were conducted and results that lead to the system functionality evaluation are presented here.
The rest of the paper is structured as follows. In section 2, we present a description of the problem and define the terms used. Then, in section 3 methodology for the development of GenMiner is presented, including the technologies that were used, and the functions that GenMiner offers are exhibited. The proposed algorithm is analytically described in Section 4, and the idea that led to this approach is depicted. In this section, the description of intermediate stages for the application of the data mining techniques is also presented. In section 5 statistical results from the conducted experiments are reported and the optimal parameters for the best possible results are also given. Finally, in Section 6 we discuss the conclusions that derive from our experiments as well as ways to possibly improve results.

2. Problem description

The basic problem we are trying to solve can be stated as follows:
"Given an amino-acid database or training set that exists in proteins with known properties (that have been experimentally specified), we aim to create a tool that can classify new, unknown proteins in some known to the training set family of proteins, referred as protein class."
In Figure 1 our approach to the problem is designated. The creation of a training set is needed in order to be able to predict the function of a protein. The training dataset should use a decision tree algorithm in order to build a decision tree. Then, in the tree evaluation step, if the results are as expected, we are able to predict an unknown protein's function.
Any protein chain can be mapped into a representation based on attributes. Such a representation supports the efficient function of data-driven algorithms, which represent instances as classified part of a fixed set of attributes. A very important issue in the data mining process is the efficient choice of attributes. In our case, protein chains are represented using a proper motif sequence vocabulary [5].
Suppose the vocabulary contains N motifs. Any given protein sequence typically contains a few of these motifs. We encode each sequence as an N-bit binary pattern where the i^{th} bit is 1 if the corresponding motif is present in the sequence; otherwise the corresponding bit is 0. Each N-bit sequence is associated with a label which identifies the functional family of the sequence (if known). A training set is simply a collection of N-bit binary patterns each of which has associated with it, a label that identifies the function of the corresponding protein. This training set can be used to train a classifier which can then be used to assign novel sequences to one of the several functional families represented in the training set. This process is illustrated in Figure 2.

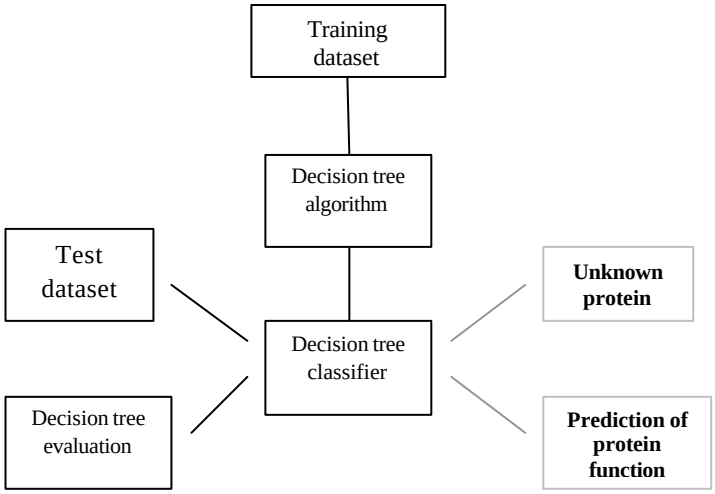


Figure 1. GenMiner's approach to the protein classification problem

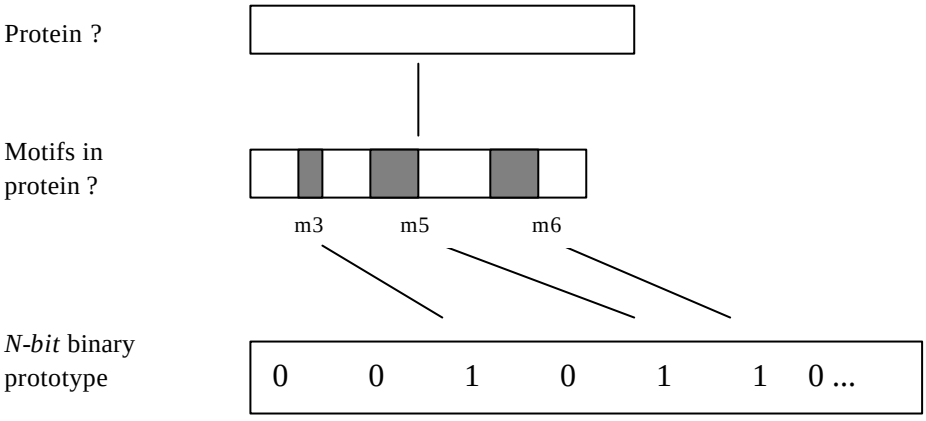


Figure 2. Protein data representation

3. GenMiner methodology and functionality

GenMiner has been developed in Java, in order to be able to communicate with the WEKA data mining module. The preprocessing procedure, as implemented in GenMiner, requires the use of database tables, thus allowing the compact representation of data. The format used in SQL Server database is used, since such kind of representation can support the use of both the SQL Analysis Manager and the WEKA data mining module. WEKA was selected as a protein data mining tool, since it provides a large variety of data mining algorithms and

decision tree building, while its open sourceness facilitates the interconnection with GenMiner.

The protein data are acquired from three major protein databases; ProSite and SwissProt's tables of strong motifs and strong prototypes. The automated procedure that GenMiner offers, in order to prepare the data for insertion in the WEKA data mining environment, provides the convenience to use any of the developed in WEKA data mining routines. In order to conduct classification experiments, the C4.5 algorithm was used. C4.5 basically chooses the threshold that yields the greatest information gained by partitioning the given training set into two subsets at that threshold. Its performance has been widely tested and is proven quiet well.

Apart from the WEKA module environment, GenMiner is able to communicate with SQL Analysis Manager as well. The Microsoft Decision Trees algorithm is the one that is used in order to classify protein data. It is based upon the notion of classification. The algorithm builds a tree that will predict the value of a column based upon the remaining columns in the training set. Therefore, each node in the tree represents a particular case for a column. The decision on where to place this node is made by the algorithm, and a node at a different depth than its siblings may represent different cases of each column. The algorithm can be parameterized in order to change its behavior when creating a model.

GenMiner preprocessing procedure results in the production of the training and test data files in WEKA or SQL format. Furthermore, it exploits DTS in order to transform the SQL files to matrix format, thus enabling the communication with SQL Analysis Manager.

From a functional point of view, GenMiner offers various services, that are presented below:

a. **Protein behavior discovering**
   The user can enter a code of a protein and select a variety of motifs contained in the protein. GenMiner discovers the family of proteins in which it belongs, along with the family's properties.
b. **Protein recognition**
   Another function of GenMiner is the capability of recognizing a given protein. Entering a protein's code, GenMiner outputs the protein's name, the motifs it contains, as well as the family in which it belongs.
c. **Decision tree building**
   Protein recognition and protein behavior discovering are achieved through the building of a decision tree. The tree remains in the user's choice to immediately study or store it. The user has also the capability to choose between two ways of building the tree, using WEKA or SQL Analysis Server.
d. **Simple and functional user interface**
   The user interface was developed in order to simplify its use and be comprehensible even to users who might not be sufficiently familiar with computer technology, such as biologists or high school students. Tooltips for each key are implemented and interaction, such as color change on mouse over are included, among others, in the graphical user interface .
e. **Integration of multiple tools in one program**
   Since our target is the development of an integrated system that can preprocess data and furthermore classify unknown proteins, the WEKA data mining platform was incorporated. In Figure 3, the dataflow diagram for GenMiner is shown. Once the user has entered a valid SQL Server protein code, he can have access to create training and test set files that can be processed both by WEKA and SQL Analyzer.

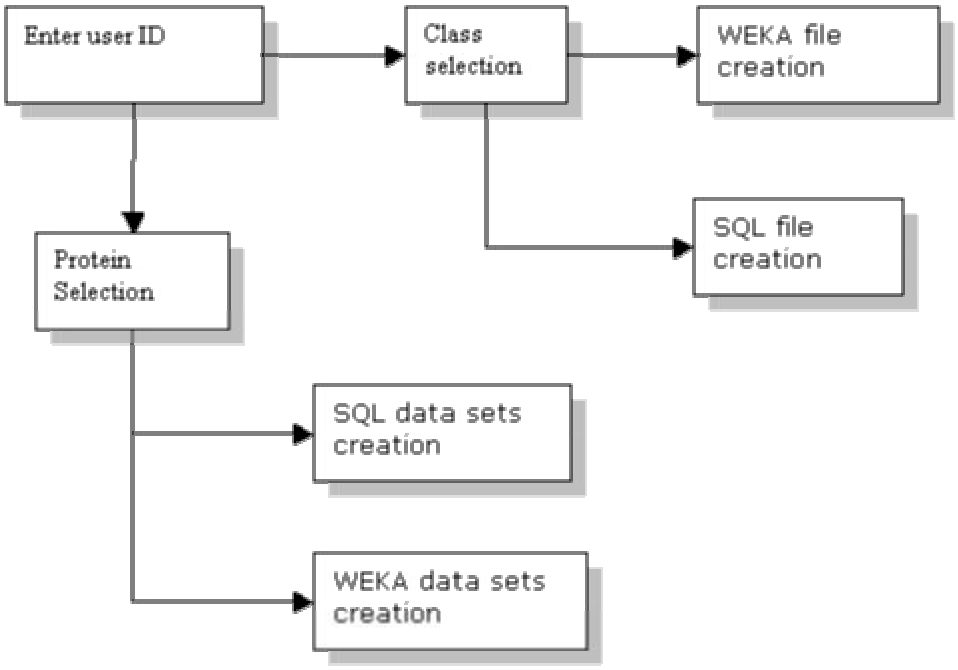


Figure 3. Dataflow diagram for GenMiner

4. GeneMiner structure

The preprocessing step is very important for the accurate representation of data. In our case, data from the Prosite database, strong prototypes and strong motifs databases had to be preprocessed. The fact that the individual protein data files are very slow to process has led us to introduce the data conversion into an environment of database tables, thus providing faster processing and robustness in the storage procedure. Therefore, the GenDatabase component was developed.

Prosite database includes more than 1100 records. Each record describes a common function in certain proteins. In the system produced, each record corresponds to a class of proteins, e,g, the record PDOC00662 corresponds to the class names "MCM family signature and profile". Prosite database records are sorted in a file, in order to ease browsing among vast quantities of data and allow programmers to develop bioinformatics tools. This file is sorted as follows:

i. Each row can contain 6-128 characters. The first two characters state the type of information that follows in the rest row.
ii. Characters 3-5 are empty, in order to separate data from their description and
iii. The rest of the characters from 6 to 128 are reserved for the row data.

A structure of the Prosite records is shown in Figure 4. We should note that the type *ID Identification* appears only once in a record and states the beginning of the record. Other types that can appear only once are *AC Accession number*, *DT Date*, *DE Short description* and *DO Pointer the documentation file*. All the remaining types may vary from 0 to infinite elements for each record.
The strong motifs database contains the analytical description of each motif, which proteins contain a certain motif, as well as the Swisss-prot correctness for each protein.

```
ID Identification
AC Accession number
DT Date
DE Short description
PA Pattern
MA Matrix/profile
RU Rule
NR Numerical results
CC Comments
DR Cross – references to SWISS – PROT
3D Cross – references to PDB
DO Pointer the documentation file
// Termination line
```

Figure 4. Structure of Prosite records

```
sw|O00628|PEX7_HUMAN|323|D405387F7F14B432 73    96
      prf|PS50082|WD_REPEATS_2|8.5|6.5|SEP-2000 11    -10
      8.169
sw|O00629|IMA4_HUMAN|521|D98BC45002C9F57E 114   158
      prf|PS50176|ARM_REPEAT|8.5|6.5|SEP-2000   1     -1
      9.327
sw|P57413|RSMC_BUCAI|338|E43516DDD22FA014 194   305
      prf|PS50193|SAM_BIND|8.5|6.5|MAY-1999     1     -1
      10.587
sw|P57415|MVIN_BUCAI|511|D95FF4B563410A9F 157   182
      prf|PS50314|PHE_RICH|8.5|6.5|SEP-2000     1     -1
      6.529
sw|P57421|FLGD_BUCAI|236|E15EAA2D3D84F293 5     38
      prf|PS50321|ASN_RICH|8.5|6.5|SEP-2000     1     -1
      6.922
sw|P57426|FLGI_BUCAI|372|CCF74D2E1B294835 296   313
      prf|PS50079|NLS BP|5.0|3.0|MAY-1999 1     -1    3.000
```

Figure 5. Sample of SwisProt's strong prototypes table

The strong prototypes database has identical properties with the strong motifs database. It contains the prototype code, the proteins that contain it, as well as their names. A sample from the strong prototypes database is given in Figure 5.
The various formats from the three databases described above was converted in a single file that can be processed by WEKA. The first line in the file contains the name of the relation whose data are represented under the tag *@relation*. The second line contains the protein known properties under the tag *@attribute*. The following lines contain the rest protein

attributes along with possible values. They are tagged under *@attribute* and their volume may vary. Finally the data rows appear under the *@data* tag, related by their attributes, values and order of appearance in each record. The WEKA format for the representation of the protein data is presented in Figure 6. For the insertion of data in the SQL Analysis Manager a DTS data transformation package was used for the conversion of the raw data in a database table.

```
@relation protein
@attribute protein
{P98140,Q04962,P00748,Q9R098,P22897,P11717,Q07113,P49260,P80964,P
81019,P81121,P02751,P04937,P02784,P04557,P08253,P33434,P33436,P52
176,P14780,P41245,Q04756,P08169,P49259,Q9UBV2,P07589,P50282,Q9061
1,P50757,O18733,P41246}
@attribute PS00023 {YES,NO}
@attribute PS50034 {YES,NO}
@attribute PS50240 {YES,NO}
@attribute PS50026 {YES,NO}
@attribute PS50070 {YES,NO}
@attribute PS00022 {YES,NO}
@attribute PS01253 {YES,NO}
@attribute PS00021 {YES,NO}
@attribute PS00134 {YES,NO}
@attribute PS00135 {YES,NO}
@attribute PS01186 {YES,NO}
@attribute PS50099 {YES,NO}
@attribute PS50231 {YES,NO}
@attribute PS50041 {YES,NO}
@attribute PS00615 {YES,NO}
@attribute PS00017 {YES,NO}
@attribute PS50079 {YES,NO}
@attribute PS50325 {YES,NO}
@attribute PS00687 {YES,NO}
@attribute PS50269 {YES,NO}
@attribute PS00024 {YES,NO}
@attribute PS00343 {YES,NO}
@attribute PS50319 {YES,NO}
@attribute PS50105 {YES,NO}
@attribute PS50276 {YES,NO}
@attribute CLASS {PDOC00022}
@data
P98140,YES,YES,YES,YES,YES,YES,YES,YES,YES,YES,NO,NO,NO,NO,NO,NO,
NO,NO,NO,NO,NO,NO,NO,NO,NO,PDOC00022
Q04962,YES,NO,NO,NO,NO,NO,NO,NO,NO,NO,YES,NO,NO,NO,NO,NO,NO,NO,NO
,NO,NO,NO,NO,NO,NO,PDOC00022
P00748,YES,NO,NO,NO,NO,NO,NO,NO,NO,NO,YES,NO,NO,NO,NO,NO,NO,NO,NO
,NO,NO,NO,NO,NO,NO,PDOC00022
Q9R098,YES,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,NO,
```

Figure 6. Protein properties representation in WEKA format.

GenMiner is also capable of searching for a specified protein, in order to include it in the training or the test set. This function requires the combination of the above explained characteristics. Specifically, simplifying the records of the Prosite file, we can derive that the protein codes, their names and the “exists in the specific class” flag lie in the DR lines. The class in which a protein belongs appears at the end of each record under the DO tag. A proper parsing algorithm has been developed in order to extract these features from the Prosite format files and represent them in a compact format. Due to the fact that the other

two files are well formed, the information required from them is easier to extract. Then, the SQL server table is produced using a
DTS package and a stored procedure. Thus, using the DTS package, two tables are created (one for each file), which include the protein code, its name, the motif or prototype code included, along with its name.
Having created the required tables, GenMiner can provide the user with all the available Prosite classes. The user can select any or all of them, in order to include them in his search. Updating the database, several basic procedures are executed. Specifically, the selected classes are entered in the database. SQL Server selects the proteins that belong in the classes selected by the user and their flag is 'T' or 'N'. The 'T' or 'N' flag indicates that the protein belongs to the specified class. A new table is created and another stored procedure inserts in the table motifs and prototypes that are included in the proteins which exist in the selected proteins table. The update function is illustrated in Figure 7. The most important procedure is the creation of the *@data* part of the derived file. In this procedure, the necessary comparisons are carried out and the data tables are constructed. In order to seek in the selected proteins and the selected motifs/prototypes tables, two cursors are used. The existence of a combination between each motif and protein is discovered by applying an SQL select query. In the case when a combination between a protein and a motif does not exist, a "NO" value is inserted in the output table, otherwise the value is set to "YES". Moreover, this procedure inserts the protein code in the beginning of each record, while adding in the end of the record the class in which the protein belongs.
The data preprocessing procedure is designed in such a way that the minimum amount of data is required to be transferred from the SQL server to the user. The selection of classes by GenMiner creates a table in the GenDatabase and this is the only transfer of data conducted until the derivation of results. Thus, the greater part of processing load falls into the SQL server, relieving the local system from the data preprocessing procedure. The application is only responsible for the correct creation of the training files and the attractive, though functional presentation of data.

4. Experiments

We have conducted several experiments with GenMiner, using WEKA and SQL Analysis Manager. We present here five of them and describe the conditions in each case, the targets and consequently the results obtained from each one of them. For space saving purposes, we present the full result table for only three out of the five experiments. For the rest of them,

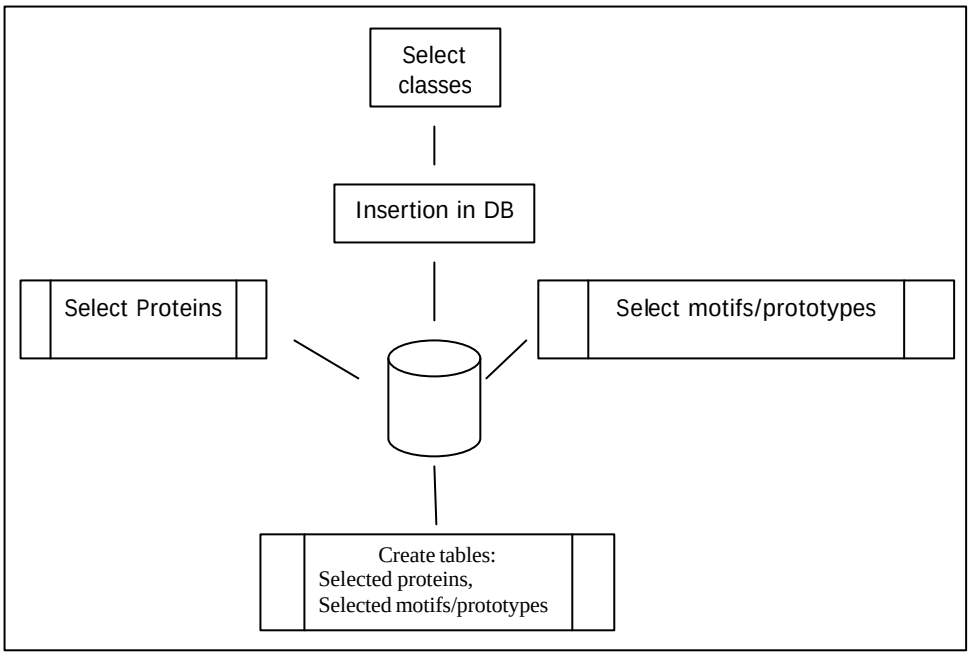


Figure 7. GenDatabase update function

only the percentage of correct classification is given. Finally, a further discussion about the results and their value follows.

4.1. Experiments using WEKA data mining.

4.1.1 Protein classification decision tree using 10 classes

In this experiment the 10 most important protein classes among all the nucleic sets were used. The selected teams, as referred in Prosite, are the following:
PDOC00662, PDOC00064, PDOC00670, PDOC50007, PDOC00154, PDOC00343, PDOC00561, PDOC00224, PDOC00791, PDOC00271.
The preprocessing procedure lasted 5:45 minutes. The output file was used for data mining using the C4.5 WEKA algorithm. The results obtained are shown in Figure 8.
During the preprocessing, a training set is exported, consisting of 1379 proteins that belong in barely 10 classes. The decision tree was produced by WEKA in 7 seconds. Using the whole training set as a test set, the percentage of successful classification was 80.13%. A view of the decision tree produced is illustrated in Figure 9.
Using GenMiner, a test set for proteins was constructed, in order to discover whether it can be properly classified. The results are shown in Figure 10. We can remark that a randomly chosen protein was correctly classified. Moreover, looking up at the confusion matrix, we can find out that the protein belongs in the class PDOC00561. In any case, GenMiner can provide the user with various useful statistics for each class separately.

4.1.2 Protein classification decision tree using 20 classes

The selected classes for this experiment were ten more than the previous one, so that the decision tree produced is bigger and possibly provides greater amount of knowledge. Thus, a greater percentage of success might be attained. The selected classes were the following:
PDOC00020, PDOC00023, PDOC00027, PDOC00335, PDOC00340, PDOC00344, PDOC00552, PDOC00561, PDOC00564, PDOC00567, PDOC00789, PDOC00790, PDOC00793, PDOC00800, PDOC00803, PDOC50001, PDOC50006, PDOC50017
Preprocessing lasted 10:13 minutes and the file was mined using the C4.5 algorithm from WEKA. The proteins used were 2174 and the full decision tree consists of 93 leaves and 185 levels. The parameters used for this experiment gave a percentage of 84.26% accuracy for all proteins. The decision tree produced is too big, thus making its illustration impossible.

4.1.3 Protein classification decision tree using 30 classes

In this experiment, 30 protein classes were used in order to study the way the results differentiate. The produced decision tree consists of 2599 proteins and 371 motifs/prototypes. The use of a greater amount of classes did not induce a proportional increase in the number of proteins. Moreover, the percentage of correct classification dropped to 74.49% , which was caused by the bigger pruning threshold that was used in this experiment. The WEKA C4.5 algorithm was proved insufficient for such an amount of data, since 43.8 seconds were required for the production of a 128 leaves and 255 levels decision tree.

4.1.4. Further discussion

Experiments using the WEKA algorithms show that the protein classification conducted by GenMiner yields a very good percentage of success along with a fast time of execution. Especially in the first experiment, the pruning threshold was at 50%, yielding very good classification results. In the second experiment no pruning was enforced, resulting in a slower time of execution, though the correct classification percentage was significantly higher. Finally, in the last experiment, due to the WEKA inability to process a sheer amount of data, pruning was much wider, yielding worse results than the previous experiments.
Experimenting with WEKA algorithms, we can conclude that, for each experiment, different

parameters should be chosen in order to succeed optimum results. In Figure 11 the optimum configuration in order to succeed the maximum percentage of correct classification and model construction speed are shown.

4.2. Experiments using SQL Analysis Manager.

4.2.1. Decision tree construction using 50 protein classes

In this experiment a decision tree for protein classification in 50 classes was constructed. GenMiner needed 22 minutes in order to preprocess data and produce a suitable file. Using DTS the data was inserted in SQL Analyzer and a model was created. Its diagram is given in Figure 12.

SQL Analyzer searches among all possible combinations, in order to discover patterns. According to the available data, the probabilities of classification in each class for each protein are calculated, thus creating a probabilistic classification tree. Using the greatest probabilities calculated, SQL Analyzer decides upon the number of leaves and levels that the produced tree should have. The user cannot modify the algorithm's sensitivity, consequently the size of the tree cannot be preset.

Each leaf of the tree contains the probability for a specified protein to belong in a certain class. In Figure 13, an overall result table is shown. We can observe the exact classification for the training set proteins.

4.2.2. Decision tree construction using 80 protein classes

The above described procedure was applied using 80 protein classes. The induced tree is illustrated in Figure 14. The color of each leaf indicates the class with the bigger probability for the specific motif/prototype. The histogram in Figure 15 shows the case where the prototype PS50071 exists in a protein's chain. Then, the protein belongs in class PDOC00033 with a probability of 43.83% and in class PDOC00027 with probability of 26.54%. In Figure 16 the classified proteins table is shown. We observe that, among the 567 proteins that contain the PS50071 prototype, 43.83% belong in the PDOC00033 class, 15.74% of the proteins belong in the PDOC00032 class, while the rest of them spread among the remaining classes.

4.2.3. Further discussion

While SQL Analyzer is a powerful commercial package, it lacks the ability to respond to the need for execution of sheer amounts of data, due to the extremely big amount of motifs/prototypes that should be processed. Even though the optimum tool for data mining was not used, the results obtained were particularly useful, fast and correct, due to the GenMiner preprocessing procedure, which made the data representation extremely compact.

5. Conclusions

We have presented GenMiner, an efficient system to mine protein data using WEKA and Microsoft SQL Analysis Manager. Preprocessing three major protein databases and combining the necessary data in a compact way, GenMiner is an integrated package which provides the data mining algorithms with enhanced reliability and robustness. The decision tree technique implemented for mining protein data has produced strong results that depict the system's capability of efficiently discovering properties of unknown proteins, while presenting them to the user in an interactive and functional interface. Our next steps involve GenMiner's capability of the data mining algorithms execution through the worldwide web, as well as the representation of data in XML format, thus enabling communication with other commercial data mining packages.

6. References-Bibliography

1. Dake Wang, Xiangyun Wang, Vasant Honavar and Drena L. Dobbs, "Data – Driven Generation of Decision Trees for Motif – Based Assignment of Protein Sequences to Functional Families", Iowa State University, Ames, IA, 2001.
2. Pericles A. Mitkas, Andreas L. Symeonidis, Dionysis D. Kehagias, "Genetic data preparation for the discovery and classification of protein families based on the protein chain", Electric and Computer Engineering Department – A.U.TH, Thessaloniki, 2001
3. Ian H. Witten, Eibe Frank, "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", Morgan Kaufmann, October 1999
4. Claude Seidman, "Data Mining with Microsoft SQL Server 2000", Microsoft Press, Redmond, Washington, 2001.
5. Amos Bairoch, Prosite, "A dictionary of protein sites and patterns – User Manual", Swiss Institute of Bioinformatics, Geneva, 1999

```
Number of Leaves  :     78

Size of the tree :      155


Time taken to build model: 6.88 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances        1105
80.1305 %
Incorrectly Classified Instances       274
19.8695 %
Kappa statistic                          0.7701
Mean absolute error                      0.0497
Root mean squared error                  0.1577
Relative absolute error                 28.5525 %
Root relative squared error             53.4411 %
Total Number of Instances             1379

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.803     0.004      0.974     0.803     0.881    PDOC00064
  0.45      0.02       0.739     0.45      0.56     PDOC00154
  0.986     0.198      0.472     0.986     0.638    PDOC00224
  0.808     0          1         0.808     0.894    PDOC00271
  0.695     0.002      0.968     0.695     0.809    PDOC00343
  0.945     0.004      0.912     0.945     0.929    PDOC00561
  0.842     0.003      0.983     0.842     0.907    PDOC00662
  0.917     0.001      0.917     0.917     0.917    PDOC00670
  0.92      0          1         0.92      0.958    PDOC00791
  0.415     0.001      0.944     0.415     0.576    PDOC50007

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 188   4  40   0   0   0   1   1   0   0 |   a = PDOC00064
   3  68  79   0   0   0   1   0   0   0 |   b = PDOC00154
   0   3 207   0   0   0   0   0   0   0 |   c = PDOC00224
   0   1  23 105   1   0   0   0   0   0 |   d = PDOC00271
   0   7  26   0  91   5   1   0   0   1 |   e = PDOC00343
   0   0   1   0   2  52   0   0   0   0 |   f = PDOC00561
   1   4  27   0   0   0 170   0   0   0 |   g = PDOC00662
   0   0   1   0   0   0   0  11   0   0 |   h = PDOC00670
   1   2  14   0   0   0   0   0 196   0 |   i = PDOC00791
```

Figure 8. Classification results using 50 protein classes

Figure 9.Constructed decision tree for 10 protein classes.

```
=== Run information ===

Scheme:       weka.classifiers.j48.J48 -C 1.0 -M 2
Relation:     protein-weka.filters.AttributeFilter-V-R2-275
Instances:    1379
Attributes:   274
              [list of attributes omitted]
Test mode:    user supplied test set: 1 instances

Time taken to build model: 6.36 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances           1               100
%
Incorrectly Classified Instances         0                 0
%
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances                1

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0         0          0         0         0        PDOC00064
  0         0          0         0         0        PDOC00154
  0         0          0         0         0        PDOC00224
  0         0          0         0         0        PDOC00271
  0         0          0         0         0        PDOC00343
  1         0          1         1         1        PDOC00561
  0         0          0         0         0        PDOC00662
  0         0          0         0         0        PDOC00670
  0         0          0         0         0        PDOC00791
  0         0          0         0         0        PDOC50007

=== Confusion Matrix ===

 a b c d e f g h i j   <-- classified as
 0 0 0 0 0 0 0 0 0 0 | a = PDOC00064
 0 0 0 0 0 0 0 0 0 0 | b = PDOC00154
 0 0 0 0 0 0 0 0 0 0 | c = PDOC00224
 0 0 0 0 0 0 0 0 0 0 | d = PDOC00271
 0 0 0 0 0 0 0 0 0 0 | e = PDOC00343
 0 0 0 0 0 1 0 0 0 0 | f = PDOC00561
 0 0 0 0 0 0 0 0 0 0 | g = PDOC00662
 0 0 0 0 0 0 0 0 0 0 | h = PDOC00670
 0 0 0 0 0 0 0 0 0 0 | i = PDOC00791
 0 0 0 0 0 0 0 0 0 0 | j =
PDOC50007
```

Figure 10. Classification of test set using 10 classes.

	Pruning	Sensitivity
Maximum classification accuracy	No	1
Maximum model construction speed	Yes	0.25

Figure 11. Optimum WEKA configuration for maximum accuracy and speed.

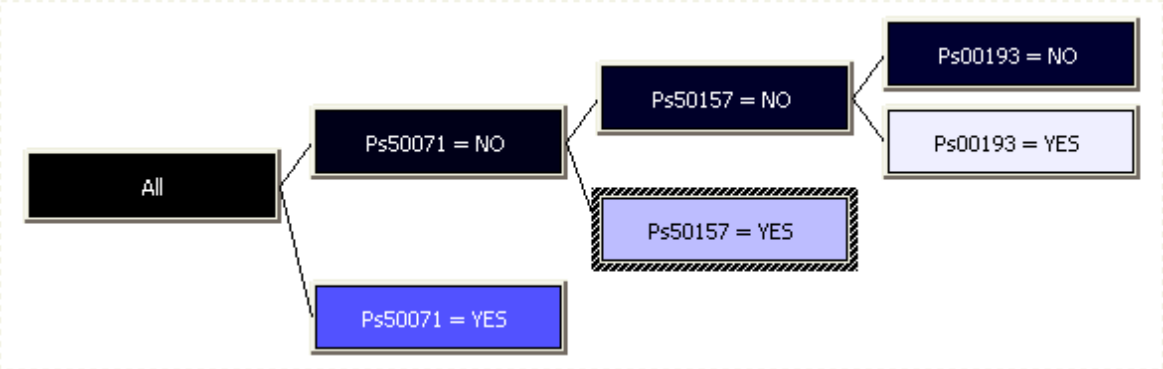


Figure 12.Decision tree for 50 classes using SQL Analyzer.

Totals | Histogram

Value	Cases	Probability
(Tree Total)	**4132**	**100.00%**
PDOC00022	94	2,27%
PDOC00023	26	0,63%
PDOC00024	47	1,14%
PDOC00026	138	3,34%
PDOC00027	335	8,11%
PDOC00028	94	2,27%
PDOC00030	160	3,87%
PDOC00031	45	1,09%
PDOC00032	112	2,71%
PDOC00033	424	10,26%
PDOC00084	67	1,62%
PDOC00085	239	5,78%
PDOC00086	72	1,74%
PDOC00087	3	0,07%
PDOC00088	45	1,09%
PDOC00089	134	3,24%
PDOC00090	6	0,15%

Figure 13 Classification results for 50 classes using SQL Analyzer

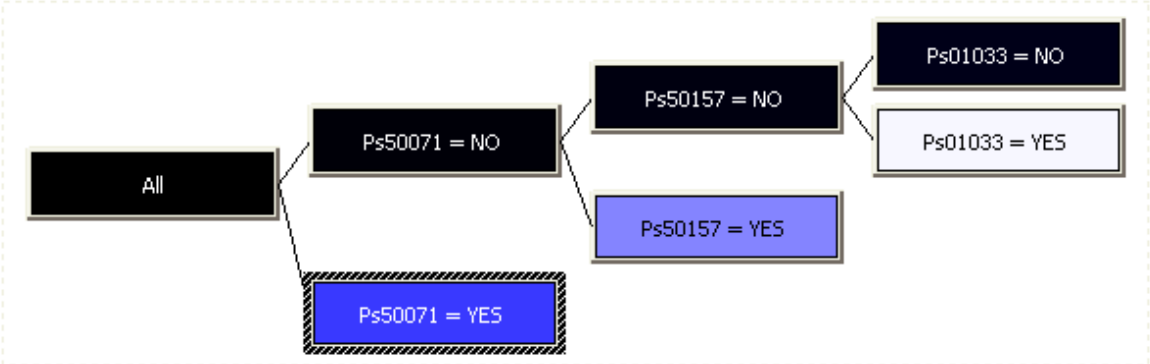
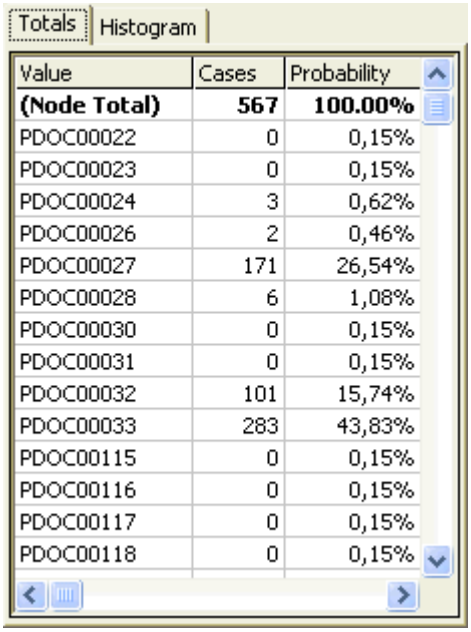


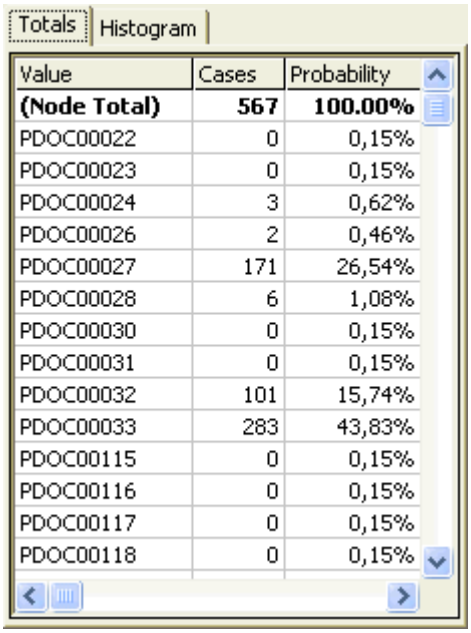
Figure 14. Decision tree using 80 protein classes



Totals | Histogram

Value	Cases	Probability
(Node Total)	**567**	**100.00%**
PDOC00022	0	0,15%
PDOC00023	0	0,15%
PDOC00024	3	0,62%
PDOC00026	2	0,46%
PDOC00027	171	26,54%
PDOC00028	6	1,08%
PDOC00030	0	0,15%
PDOC00031	0	0,15%
PDOC00032	101	15,74%
PDOC00033	283	43,83%
PDOC00115	0	0,15%
PDOC00116	0	0,15%
PDOC00117	0	0,15%
PDOC00118	0	0,15%

Figure 15. Histogram analysis for the prototype PS50071 in 80 class classification



Totals | Histogram

Value	Cases	Probability
(Node Total)	**567**	**100.00%**
PDOC00022	0	0,15%
PDOC00023	0	0,15%
PDOC00024	3	0,62%
PDOC00026	2	0,46%
PDOC00027	171	26,54%
PDOC00028	6	1,08%
PDOC00030	0	0,15%
PDOC00031	0	0,15%
PDOC00032	101	15,74%
PDOC00033	283	43,83%
PDOC00115	0	0,15%
PDOC00116	0	0,15%
PDOC00117	0	0,15%
PDOC00118	0	0,15%

Figure 16. Classification results for 80 classes using SQL Analyzer