A System for Managing Experiments in Data Mining

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Greeshma Myneni

August, 2010
A System for Managing Experiments in Data Mining

Greeshma Myneni

Thesis

Accepted: 

Approved: 

Advisor  
Dr. Chien-Chung Chan

Dean of the College  
Dr. Chand Midha

Committee Member  
Dr. Kathy J. Liszka

Dean of the Graduate School  
Dr. George R. Newkome

Committee Member  
Dr. Yingcai Xiao

Date

Department Chair  
Dr. Chien-Chung Chan
ABSTRACT

Data Mining is the process of extracting patterns from data. There are many methods in Data Mining but our research mainly focuses on the classification methods. We present the existing data mining systems that are available and the missing features in those systems. An experiment in our research refers to a data mining task. In this research we present a system that manages data mining tasks. This research provides various advantages of managing the data mining tasks. The system to be dealt with in our research is the “Rule-based Data Mining System”. We present all the existing features in the Rule-based Data Mining System, and show how the features are redesigned to manage the data mining tasks in the system. Some of the new features are managing the datasets accordingly with respective to the data mining task, recording the detail of every experiment held, giving a consolidated view of experiments held and providing a feature to retrieve any experiment with respect to a data mining task. After that we discuss the design and implementation of the system in detail. We present also the results obtained by using this system and the advantages of the new features. Finally all the features in the system are demonstrated with a suitable example. The main contribution of this thesis is to provide a management feature for a data mining system.
TABLE OF CONTENTS

LIST OF FIGURES........................................................................................................ vii

CHAPTER

I. INTRODUCTION........................................................................................................ 1
  1.1 Machine Learning ......................................................................................... 1
    1.1.1 Learning Strategies............................................................................... 1
    1.1.2 Inputs and Outputs............................................................................... 2
    1.1.3 Testing................................................................................................. 4
  1.2 Tools............................................................................................................... 5
    1.2.1 WEKA.................................................................................................. 5
  1.3 Observations.................................................................................................. 6
  1.4 Proposed Work.............................................................................................. 7
  1.5 Organization of the Thesis.......................................................................... 8

II. FEATURES OF EXPERIMENT MANAGEMENT SYSTEM .......................... 9
  2.1 Introduction.................................................................................................. 9
    2.1.1 Upload................................................................................................ 10
    2.1.2 Learn.................................................................................................. 10
    2.1.3 Test..................................................................................................... 11
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Decision Tree for Playing Tennis</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Rules of Decision Tree for Playing Tennis</td>
<td>4</td>
</tr>
<tr>
<td>3.1</td>
<td>ER Diagram</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>Database Design for Managing Experiments</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Sample Attribute File</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>Sample Data File</td>
<td>24</td>
</tr>
<tr>
<td>4.3</td>
<td>Upload Snapshot</td>
<td>25</td>
</tr>
<tr>
<td>4.4</td>
<td>Learn Snapshot</td>
<td>26</td>
</tr>
<tr>
<td>4.5</td>
<td>Generate Test File Snapshot</td>
<td>28</td>
</tr>
<tr>
<td>4.6</td>
<td>Test Snapshot</td>
<td>29</td>
</tr>
<tr>
<td>4.7</td>
<td>Learn and Test Snapshot</td>
<td>30</td>
</tr>
<tr>
<td>4.8</td>
<td>Experiment Snapshot</td>
<td>32</td>
</tr>
<tr>
<td>5.1</td>
<td>Attribute File for Bench Dataset</td>
<td>37</td>
</tr>
<tr>
<td>5.2</td>
<td>Data File for Bench Dataset</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>Experiment Snapshot after Upload of Dataset</td>
<td>39</td>
</tr>
<tr>
<td>5.4</td>
<td>Experiment Snapshot after Learning</td>
<td>40</td>
</tr>
<tr>
<td>5.5</td>
<td>Experiment Snapshot after Generating a Test file</td>
<td>41</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

1.1 Machine Learning

Learning is important for practical applications of artificial intelligence. According to Herbert Simon [1], learning is defined as “any change in the system that allows it to perform better the second time on repetition of the same task or on another task drawn from the same population”. The main objective of machine learning methods is to extract relationships or patterns, hidden among large pile of data. The most popular machine learning method is learning from example data or past experience. The example data is also called as training data. Machine learning has many successful applications in fraud detection, robotics, medical diagnosis, search engines etc [1, 2].

1.1.1 Learning Strategies

There are two main categories in machine learning: supervised learning and unsupervised learning. Classified training data has a decision attribute along with condition attributes. The supervised learning classifier generates rules, using the classified training data [2]. The rule is a simple model that explains the data and fits the entire data.
In unsupervised learning, the data is not classified. The main objective of this learning is to find the patterns in the input [2]. One form of unsupervised learning is clustering, where the aim is to group (cluster) the input data. There are many other types of machine learning, which can be referred from [5, 3].

1.1.2 Inputs and Outputs

In this thesis, we are mainly interested in supervised learning. The input given to the classifier is classified training data. The training data is composed of input and output vectors. The input vector is characterized by a finite set of attributes, features, components [2]. The output vector is also called a label, or a class, or a category or a decision. The input and output vectors can be of real valued numbers, discrete valued numbers and categorical values, which are finite sets of values. The training data may be reliable or may contain noise [5]. Data with missing values complicates the learning process. Hence before input is given to the machine learning system preprocessing is needed. Data pre-processing [4] includes cleaning, normalization, transformation, feature extraction and selection. Typical input to the learning system can be a text file containing all training examples. In general, the input has two files namely data file and attribute file.

When the input is given to the learning system, the learning algorithm generates the rule set. The rule set generated might not be perfectly consistent with all the data, but it is desirable to find a rule set that makes as few mistakes as possible. The representation of learned knowledge varies with the learning system. Figure 1.1 gives the representation of learning in a decision tree.
In decision tree learning [6], the representation of the learned knowledge is represented using decision trees. The classifications are represented by the leaf nodes. The collections of features that derive to these classifications are represented by branches. An unknown type is classified by traversing the entire tree and taking appropriate branch. This continues until a leaf node is reached.

Figure 1.1 shows an example of decision tree for play tennis. In Figure 1.1[5, 9], the classifications are yes and no. The internal node indicates the property and branches test the individual value for that property.

![Decision Tree for Playing Tennis](image)

**Figure 1.1 Decision Tree for Playing Tennis**

From the decision tree in Figure 1.1 we can generate rules as shown in Figure 1.2 [5, 9]
If outlook is sunny, and humidity is high then don’t play tennis.

If outlook is sunny, and humidity is low then play tennis.

If outlook is overcast, then play tennis.

If outlook is rain, and wind is strong then don’t play tennis

If outlook is rain, and wind is weak then play tennis.

Figure 1.2 Rules of Decision Tree for Playing Tennis

1.1.3 Testing

Testing helps to validate the learned knowledge, and calculate the performance of the classifier. There are many testing strategies applied in order to validate. Some of the popular testing strategies are sub sampling and N-fold Cross Validation [9].

In the sub sampling method, the dataset is split into training data and validation data. For each split, the training is done on training data and tested across the validation data. The results are then averaged over the splits. The main disadvantage of this method is some observations may not be selected or some observations may be selected more than once. In other words, validation subsamples may be overlapped.

The other common method type for testing is N-Cross Validation. In N-Cross Validation, the dataset is partitioned into N-1 equally sized subsamples. Each subsample is used as the test set for a classifier trained on the remaining N-1 sub samples. This process is repeated N times, and the average accuracy is calculated from these N folds.
The main metric in calculating the accuracy of a supervised learning classifier is the percentage of correctly classified instances. By applying these testing methods, we can understand the performance and accuracy of the classifier for the particular dataset. The goal of these testing methods is to obtain a rule set that is independent of the data used to train the dataset.

1.2 Tools

Some of the tools available to perform machine learning experiments are WEKA [3, 13], C4.5 [9] and Pentaho [7]. Among the tools available the popular tool in the area of data mining and machine learning is WEKA. The following sections give a brief overview of this tool and examine issues and the proposed work to overcome those issues.

1.2.1 WEKA

There are many tools which support data mining tasks. Waikato Environment for Knowledge Analysis abbreviated as WEKA [3, 12] is a popular collection of machine learning software written in Java which is developed at the University of Waikato. WEKA [3] is a collection of machine learning algorithms for data mining tasks. From [18], “WEKA supports several standard data mining tasks like data preprocessing, clustering, classification, regression, visualization, and feature selection”. The main user interface in WEKA is Explorer. There is also another interface namely Experimenter which helps in comparing the performance of WEKA’s machine learning algorithms on group of datasets.

The graphical interface to WEKA’s core algorithms is available through Knowledge Flow [3, 13] which is an alternative to Explorer. In Knowledge Flow [19] the
data is processed in batches or incrementally. There are different components in Knowledge Flow, some of them are TrainingSetMaker, TestSetMaker, CrossValidationFoldMaker, TrainTestSplitMaker etc. It helps in processing and analyzing the data. The different tools in WEKA like classifiers, filters, clusterers, loaders, savers and with some other tools are available in Knowledge Flow.

1.3 Observations

There are many data mining tools available like Pentaho[7], Oracle[14], Microsoft SQL Server[15, 16] etc.

One of the open source related data mining engines is Pentaho. Pentaho[7] is a collection of tools for machine learning and data mining. It is a set of different data mining techniques like classification, regression, association rules and clustering. Pentaho is based on WEKA data mining and is tightly integrated with core business intelligence capabilities.

Microsoft SQL Server [15] provides many features in the area of data mining and making predictive analysis. It is integrated within the Microsoft Business Intelligence platform and extends its features into business applications. Oracle Data Mining [14] provides a wide set of data mining algorithms which help in solving business problems. Access to Oracle Database also has access to Oracle Data Mining. Oracle Data Mining also helps in making predictions and using reporting tools which include Oracle Business Intelligence EE Plus.

These tools help in performing data mining tasks and making predictive analysis, but this analysis is made in a single data mining task. In reality, many data mining tasks are performed on a single data set, when there are multiple data mining tasks it is
necessary to compare the results with other tasks and manage them accordingly. The accuracy and results among the data mining tasks differ, by having a management system in data mining it would help in making analysis much easier and thereby to take decisions.

1.4 Proposed work

In this thesis, an experiment refers to a data mining task. An experiment can be uploading a dataset, learning from dataset, performing testing, or learning and testing from dataset. A typical experiment can be learning from dataset and testing on the generated rule set. To perform, learn, and test the inputs or parameters are number of training and test files are to be generated dynamically from the data set and the split by which they are generated. The details of different experiments involved are discussed in Chapter II.

In machine learning algorithms, we try to perform many experiments to get the most possible patterns or results, so it is equally important to manage those experiments. We use many datasets, and we might perform many experiments on the same dataset. It is necessary to manage the datasets accordingly with respect to the raw data, learned data, test data etc. Management of experiments implies managing the datasets accordingly, recording the experiments held and the results systematically. By providing this feature it reduces time in conducting the number of experiments.

From the above observations and the background, it is necessary to build a system for management of experiments in Rule-based Data Mining System [10, 17]. The main objective of this thesis is to provide a feature for management of experiments, design and implement the features and validate the implementation in Rule-based Data Mining
System. The Knowledge flow component in WEKA is similar to the Rule-based data mining system. It has similar features like the features in Knowledge flow component in WEKA. By adding these features it gives an intuitive idea to the user the experiments need to be held i.e., the parameters that are to be changed for the desired results. Some of the features are implemented by following the work flow management standards like identifying the dependencies, designing the abstract level initially [21, 23]. The proposed work would help the user in using the features easily and organizing the experiments orderly.

1.5 Organization of thesis

This thesis covers the development of a system for managing experiments in Web based Machine Learning Utility. This thesis is organized as follows:

Chapter II describes the features of experiment management system.

Chapter III focuses on the design for developing this system. The database design, ER Model of the system is discussed in detail.

Chapter IV describes the implementation of the system, and how it is been implemented along with a detailed description of the interface.

Chapter V explains the overall evaluation of the system with the test cases.

Finally Chapter VI presents a summary of the work done in this thesis. It also summarizes additional functionalities that were developed in this thesis and concludes with future work.
CHAPTER II

FEATURES OF THE EXPERIMENT MANAGEMENT SYSTEM

The experiment management system is the system for managing the data mining tasks. The main objective of this system is to manage all the data mining tasks mentioned above. In real time, the numbers of experiments increase rapidly and analysis is done for each experiment, to obtain desired results and accuracy. Depending upon the analysis, the experiments are carried out by changing different parameters. Thus, a machine learning experiment requires more than a single learning run; it requires a number of runs carried out under different conditions [8]. So, there is a need to manage these experiments accordingly, thereby giving a detailed view of the experiments and giving an intuitive idea to the user what experiments need to be held for better results. This chapter gives brief introduction of the system, the data mining tasks involved and a brief description of new features in the experiment management system.

2.1 Introduction

The system to be managed in our research is the “Rule-based Machine Learning Utility”. This utility focuses on learning from examples using the BLEM2 learning algorithm. BLEM2 implies learning Bayes rules from examples. Following sections describe the features in this utility.
2.1.1 Upload

The upload operation is used to upload files from the local computer to the system. The user has an option to select different formats, but our main consideration is BLEM2 format. The BLEM2 takes categorical data as input. BLEM2 accepts two files a data file and an attribute file wherein attribute file contains information about attributes where as data file contains actual data. The details of the format of the input are discussed more in detail in Chapter IV (Implementation).

2.1.2 Learn

The learn operation is used to generate rules. With the supplied input and attribute data, the BLEM2 algorithm generates rules. The BLEM2 program [17] gives out eight files as output namely the Certain Rules file, the Possible Rules file, the Boundary Rules file, the BCS file, the Stats file, the Textfileoutput file, the Output file and the Nbru file. This BCS file has the certainty factor, coverage factor and strength factor. The rules learned from the lower approximation set are called certain rules, rules learned from the upper approximation set are called possible rules, and rules learned from the boundary set are boundary rules.

The certainty factor denotes the ratio of the covered examples with the same decision value of the rule. The strength factor is the support of the rule over the entire training set, and the coverage factor is the ratio of the decision value class covered by the rule. The generated rules can be used to predict the decision values for new examples. There are many strategies, by which the decision weights are computed from the rules.
The weights are calculated in four ways, certainty * coverage, certainty * strength, certainty alone and coverage alone.

2.1.3 Test

The test operation is used to test the rules generated by the learn operation. The user can upload their own test file and test the rules on the rules generated. The test file contains the examples, the same as a data file without a decision value. If the user doesn’t have a test file, a random test file can be generated by the system. The input is the split i.e., the percentage of examples that are randomly taken from the train data. The test file is generated by taking random examples from the train data depending upon the split. The generated test file can be tested on different rule data, but similar input data.

2.1.4 Learn and Test

The “learn and test” operation is used to combine the two operations learn and test. In this operation it takes two parameters as input, firstly the number of training and testing files to be generated and the split by they should be generated. For example if the split is 20%, the train data takes randomly 80% of examples from the original input data; the remaining examples are stored as the test data. This procedure is repeated for n number of iterations, where n is the number of training and testing files to be generated. The generated train and test data are saved with respective to iteration.

For all iterations, the rules are generated for the respective training data. The generated rules are also saved with respective to iteration. By selecting the weight calculation method and matching criteria the rules are tested upon the respective test data thereby calculating the confusion matrix and accuracy [17]. The confusion matrix is used
for calculating the accuracy of the classification system. A confusion matrix [20] is represented in the form of matrix which contains information about the actual classifications and the classifications predicted by a classification system.

The result contains eight files, cru, pru, bru, bcs, nbru, sts, textfileoutput and out. For all iterations the results are summarized respectively. The results can be viewed and downloaded by selecting iteration.

2.2 Experiment Management System

The experiment management system manages the Rule-based Machine Learning Utility. The features are redesigned such that they can be managed in a much easier manner, and take advantage of all the tasks that have been performed already. The following sections give detailed features of the experiment management system.

2.2.1 Upload

Each mentioned feature in the utility can be used only once on one particular dataset at a time. But however, in real time we might want to operate on multiple datasets simultaneously and correlate the results accordingly. Hence we need to save the dataset each time the new data set is uploaded, and can be referenced in the future. A unique dataset name is prompted for while uploading the dataset, and is written to the database. All future data mining tasks are referenced by this data set name.

2.2.2 Learn

In the learn feature, all the datasets which are uploaded are populated. The user can select the dataset and learn the rules from the selected dataset. The datasets which are haven’t learned, are only populated in this feature, so that it doesn’t generate rules again and again on the same datasets which have already generated rules. As we have observed...
in the learn feature of Rule-based Machine Learning Utility, the learn system generates
eight different files. All the rule data files are also saved to the database.

2.2.3 Generate Test File

This is the new feature in this system. The dataset can be selected dynamically
from the existing uploaded datasets. This feature is used to generate a test file from the
dataset. The input is the split i.e., the percentage by which it should randomly select from
the dataset. The generated test file is saved to the database for future testing.

2.2.4 Test

The test feature is used to test the rules. The test file can be selected from the
generate test file or it can be uploaded from the local disk if the user has the own test file.
The datasets on which rules are generated are populated for selection, since without
learning data, the test feature cannot be used. The dataset and test file can be dynamically
selected from the uploaded files. The results after testing the rules are saved to the
database.

2.2.5 Learn and Test

In learn and test, the dataset can be selected from the uploaded datasets. In this
feature all datasets are populated. Once the dataset is selected, the training testing files
are generated, the rules are learned and the rules are tested, the results are saved to the
database, along with the inputs to the learn experiment i.e., the number of training and
testing files, the split, the matching criteria and the weight calculation method are all
saved to the database.
2.2.6 Experiments

Each data mining task that is performed is saved and referenced as an experiment. The management system records all those experiments that are performed are stored in a precise manner. The experiment gives the detailed information about the dataset involved, the operation performed, and the results with respect to each experiment. Depending upon the experiment the results are shown relative to the experiment.

Each experiment has two options: delete or download. The experiment can be deleted anytime. The user has an option to download the results from the experiment. When the user selects the download option, all the results performed in the experiment are zipped and prompted to save in the local disk. The experiment feature helps to manage all the experiments very easily and gives a consolidated view in detail of the experiments performed. It helps the user to easily analyze the results and make decisions.

Chapters III and IV give the details of the design and the implementation of the system in detail.
CHAPTER III

DESIGN

In this chapter a detailed design of the experiment management system is discussed. The design consists of the ER Model and the respective database design. The typical work flow in this system is initially to upload a dataset, learn the rules from the uploaded dataset, perform any learn and test or test only experiments, and finally the results can be viewed or downloaded accordingly.

3.1 ER Model

The diagram in Figure 3.1 is the complete Entity Relationship diagram which presents the abstract, theoretical view of the major entities and relationships for experiments. Most of the entities and relationships in the Figure 3.1 are straight forward and can be easily understood. The main entities identified are rawdata, ruledata, testdata, experimentdata.

Raw data contains all information about the data and attributes of the dataset. Given rawdata when it is has learned a unique rule set is generated and stored in ruledata. To learn, the files should be of BLEM2 type. The relationship is one to one because unique raw dataset generates a unique rule dataset.
The raw dataset when it is learned and tested it is stored in the *ruledata*. Each time this operation can be experimented with different parameters like number of training and testing files etc, hence it is a one to many relationships. The *ruledata* can also be tested with the test file the resultant set is stored in *testdata*, each *ruledata* can be tested by different test files for desired results, hence it is a one to many relationship. These are all the experiments performed for necessary results. Each experiment performed is recorded in the *experimentdata*, the test files are also stored in this entity. Each experiment is unique accordingly hence it is a one to one relationship with all entities.
3.2 Database Design

The ER diagram in figure 3.1 gives the outline of all the entities and relationships. For further understanding, below Figure 3.2 shows the detailed database design.
Notations in a Database Diagram:

**Endpoints** The endpoints of the line indicate whether the relationship is one-to-one or one-to-many. If a relationship has a ‘1’ at one endpoint and a ‘*’ at the other, it is a one-to-many relationship. If a relationship has a ‘1’ at each endpoint, it is a one-to-one relationship.

**Line Style** The line indicates that there is a relationship between the tables. For every instance in one table there is a relationship to the table by which arrow points.
Figure 3.2 Database Design for Managing Experiments in Rule-based Machine Learning Utility
3.2.1 Tables

As we can see above, there are five tables each for their respective purpose. All tables are described below:

**tblRawData**  This table is used to store the raw data. Raw data implies the data and attribute files which are used to learn, test or learn and test. For each set of data and attribute files we give a name to the dataset and stored as the tablename which is given by the user. This table also stores the delimiter by which the files are delimited and also the learning type i.e., either blem2 or arff or c2 algorithms. It also stores the date on which the files are uploaded.

**tblRuleData**  After applying the learning algorithm to the raw data we obtain the rules, these rules are stored in the form of files. So these files are uploaded to the tblRuleData. The table tblRuleData stores the rule data of a particular dataset stored in the tblRawData. The ruledata is identified by the dataset name. It also stores the learning type by which it has been learned. The date on which it is learned is also stored in the table.

**tblTestFile**  This table stores the test files. Given the rule data if testing is to be performed we need a test file for a particular dataset. This test file is uploaded by the user, these are uploaded to the table tblTestFile. So, the dataset associated with the test file is stored in the table. The date when it is uploaded is also stored.

**tblTestData**  To store the information regarding test or learn and test we use this table. To perform the test operation, the learning has to be performed first. Once learning is done, this information which is stored in tblRuleData acts as an input and performs test operation and store in tblTestData. Along with this for the test operation we need a test
file which has to be uploaded in the tblTestFile. Having these files, test can be performed. The details stored in the tblTestData are dataset name, testfile, parameters used for performing the test operation.

The details when the learn and test operation is performed is also stored in the tblTestData. To perform this operation the dataset in the tblRawData is used. It stores the information of the split used to split the dataset into train and test, and the number of such sets were generated, learned and tested. It also stores the matching criteria and weight calculation method used to perform this operation. Along with these details the date when the test operation is performed is also stored.

**tblExperimentData** For every operation that was performed it stores the summary of the instance. The user can view all the operations being performed and if the user wants to retrieve the files accordingly can be retrieved by relating with the other tables.

3.2.2 Relationships

The relationship between tblRawData and tblRuleData is a one to one relationship. For a unique tblRawData when learned, a unique rule data is stored in tblRuleData. The relationship between tblRawData and tblTestData is one to many relationship, given a raw data many combinations of learning and testing experiments can be performed. The relationship between tblRuleData and tblTestData is a one to many relationship. The tblRuleData with the input of test file many testings can be performed, and the resultant results are stored in the tblTestData.
The relationship between each table with tblExperimentData is a one to one relationship.

For every operation performed the tblExperimentData has the reference of the operation.
CHAPTER IV

IMPLEMENTATION

This chapter gives a detailed implementation of the experiment management system. The implementation includes the input to the system, extends its explanation of how the existing features are redesigned to manage the data mining experiments with the snapshots at each feature. The implementation is written in the C# ASP.NET Programming Language, with Microsoft SQL Server as the back end.

4.1 System Input

The application takes two files as input. One file contains the information about the attributes; the other file provides details about the data. We will use a sample dataset to illustrate the format of these two files.

<table>
<thead>
<tr>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
<tr>
<td>Size c 3</td>
</tr>
<tr>
<td>small medium big</td>
</tr>
<tr>
<td>Color c 4</td>
</tr>
<tr>
<td>yellow green red blue</td>
</tr>
<tr>
<td>Feel c 3</td>
</tr>
<tr>
<td>soft moderate hard</td>
</tr>
<tr>
<td>Material c 3</td>
</tr>
<tr>
<td>plastic metal wood</td>
</tr>
<tr>
<td>Attitude c 2</td>
</tr>
<tr>
<td>positive negative</td>
</tr>
</tbody>
</table>

Figure 4.1 Sample Attribute File
In the Figure 4.1, the first integer indicates the position of decision attribute, and the second integer indicates the number of attributes. Information of each attribute is defined in two lines. One line contains attribute name, types of values, and number of values. For instance, the third line contains: Size is the attribute name; ‘c’ denotes categorical values; 3 denotes three values. The next line contains the symbolic values of the Size attribute such as small, medium and big. The metadata file can be either comma separated or space separated.

```
big yellow soft plastic positive
small green soft plastic negative
small yellow soft plastic positive
small yellow soft plastic positive
small red soft plastic positive
small yellow moderate plastic negative
big yellow soft plastic positive
big yellow soft plastic positive
small yellow hard metal negative
small blue soft wood negative
small yellow hard wood positive
small yellow hard wood negative
small yellow hard wood negative
medium blue soft wood negative
```

Figure 4.2 Sample Data File

In Figure 4.2, each line denotes one training example, separated by space. Information of each attribute corresponds to the attribute in the attribute file and is identified by a delimiter accordingly. Below section describes how the files are uploaded to database.

4.1.1 Upload

To upload the data file and attribute file from the client side to the server, the user will first choose “Upload” menu and provide the dataset name and choose the files on his/her local machine to upload, the delimiter by which the data is separated in the files.
Figure 4.3, shows the upload snapshot. The interface has also upload help which provides the detailed information of the files that are accepted.

Once the upload button is clicked, the system will check the uniqueness of the dataset name whether the dataset already exists in the database. Depending upon the delimiter chosen, it parses the data in the data file and attribute file and stores the files onto local disk. Later, the files are stored into database; precisely these files are stored into tblrawdata table. The dataset name, delimiter, the type of a file, file size and the date are recorded.
Appendix A, contains the function which directly writes the files into the database, tblrawdata table.

4.2 System Output

4.2.1 Learn

To obtain the rules for any dataset, the user can browse the learn menu. In the learn menu, all raw datasets are displayed i.e., the datasets which rules aren’t generated. Figure 4.4, shows a learn snapshot. To generate a ruleset for a particular dataset, select the rawdataset and click the button Learn using BLEM2.

![Figure 4.4 Learn Snapshot](image)

When the button is clicked, it gets the data and attribute file of that particular dataset from tblrawdata, learns the data, generates rules and writes the rules to the local disk. The files (ruleset) that are written to the local disk are written to the database;
The dataset name for which the rules are generated, the method used to generate the rules, filename (rule), file size and the date on which the rules are generated are stored respectively.

The function to write the rules to the database is shown in Appendix A; only the parameters change accordingly if it is a raw data set or rule dataset. Once the rules are generated for particular dataset, the dataset doesn’t appear in the learn menu, since it is already learned.

The ruledataset that has been generated can be used for further testing. Once rules are generated the ruleset can be used as many times without generating rules for each testing operation.

4.2.2 Generate Test File

To generate a test file, from a raw data set the user can browse through “generate testfile” menu. In this menu, two options are shown, the dataset and the split by which the testfile is generated. Figure 4.5, shows the snapshot.

By clicking the “Generate” button it uses a random function and picks the data, stores temporarily onto local disk. After completion the file is written to the database; precisely to the tblTestFile table, it stores the dataset name, split and the date when the test file is generated. Similar to the function in Appendix A, a function writeToTestfile is written storing all the corresponding parameters.

The testfile generated can be used later for testing. This test file could be used for the same dataset or the similar type of dataset. If the user doesn’t have a test file then this menu can be used for generating any number of test files.
4.2.3 Test

To perform testing the user can choose the “test” menu. To use this menu the user has to generate rules. If the rule set is empty then this menu cannot be used. In this menu, all the rulesets are displayed. The user must choose a ruleset, and browse a test file. If the user browse his own test file, the testfile is recorded in tblTestFile for future use. If the user doesn’t have any test files then the user can generate test file from “generate testfile” and then use this menu. All the generated testfiles are also displayed.

The matching criteria and weight calculation method have to be selected for further testing. Figure 4.6 shows the snapshot.
Once the button “Test the rules” is clicked, the accuracy and the confusion matrix are generated and stored in a file onto local disk. Later files are stored to the database; precisely tblTestData. In this table every detail is recorded, the rule dataset, the testfile used, the matching criteria, the weight calculation method and the date on which the testing is performed. The function similar to Appendix A is written for writing the testdata called writetoDBTestdata.

4.2.4 Learn and Test

The learn and test option is chosen when the user wants to learn and test immediately the user can browse “learn and test” menu. The user has to select a raw
dataset, number of training and testing files that need to be generated, learned and tested, the split by which the testing files has to be generated, the matching criteria and the weight calculation method. Figure 4.7 shows the snapshot of Learn and Test.

Figure 4.7 Learn and Test Snapshot

“GLT” implies generate, learn and test, since all operations are performed at a single click. Once the GLT button is clicked all operations are performed and the results are stored in respective files onto local disk.

The files are then stored in the database; in the tblTestData table. In this operation every detail is recorded; the dataset, the number of iterations, the split, matching criteria, weight calculation method and the date.
4.2.5 Experiment

The experiment is the most important feature in this utility. This feature helps in managing the experiments in Rule-based Data Mining System. As we have observed there are different experiments involved, so these experiments rapidly increase as time goes by. The datasets increase and the experiments on the datasets also increase day by day; therefore for analysis and desired results and accuracy these experiments are managed. Management of Experiments implies for each data mining task performed a reference of the task along with results are recorded. Each experiment has the field operation, depending upon the operation it is referenced by its respective table.

After each experiment the summary of each operation is written to the database; precisely to the table tblExperimentData. A function in the Appendix B is called the end of each operation and written to the table “tblExperimentData”. This function is records the details of every experiment. Following sections describe how the experiments are recorded for each data mining task. Figure 4.8, summarizes the experiments performed.
Let us consider each data mining task, its results and how they are managed. First, to perform any type of experiments we need to upload a data set. The experiment is named as “upload”, and is recorded with its name, the name given to the dataset when uploaded. Throughout the data mining tasks performed on this data set are mainly referenced by its dataset name. Hence it is equally important to give the proper name to the dataset when it has been uploaded. The details of the dataset are very important in performing any data mining tasks or viewing the related experiments on the tasks, hence it is recorded. As we can see in Figure 4.8 the first experiment is the upload operation referred by expid “2”, and we can observe the date that has been recorded, the name of

<table>
<thead>
<tr>
<th>Date</th>
<th>Expid</th>
<th>Tablename</th>
<th>Operation</th>
<th>Testfilename</th>
<th>Split</th>
<th>Null</th>
<th>Matching</th>
<th>WeightCalc</th>
<th>Learntype</th>
<th>iteration</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1/12/2010</td>
<td>2</td>
<td>m26</td>
<td>upload</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>Mem2</td>
<td>NA</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>3</td>
<td>m26</td>
<td>load</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>Mem2</td>
<td>NA</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>4</td>
<td>m26</td>
<td>generate</td>
<td>test</td>
<td>20</td>
<td>1</td>
<td>NA</td>
<td>Mem2</td>
<td>NA</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>5</td>
<td>m26</td>
<td>test</td>
<td>m26test20</td>
<td>20</td>
<td>1</td>
<td>ExactMatch</td>
<td>Iteration²</td>
<td>Iteration²</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>6</td>
<td>m26</td>
<td>load and</td>
<td>test</td>
<td>NA</td>
<td>20</td>
<td>1</td>
<td>ExactMatch</td>
<td>Iteration²</td>
<td>Iteration²</td>
<td>Delete</td>
<td>Download</td>
</tr>
<tr>
<td>1/12/2010</td>
<td>7</td>
<td>initial</td>
<td>upload</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>Mem2</td>
<td>NA</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>8</td>
<td>initial</td>
<td>load</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>Mem2</td>
<td>NA</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>9</td>
<td>initial</td>
<td>generate</td>
<td>test</td>
<td>20</td>
<td>1</td>
<td>NA</td>
<td>Mem2</td>
<td>NA</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>10</td>
<td>initial</td>
<td>test</td>
<td>m26test20</td>
<td>20</td>
<td>1</td>
<td>ExactMatch</td>
<td>Iteration²</td>
<td>Iteration²</td>
<td>Delete</td>
<td>Download</td>
<td></td>
</tr>
<tr>
<td>1/12/2010</td>
<td>11</td>
<td>initial</td>
<td>load and</td>
<td>test</td>
<td>NA</td>
<td>15</td>
<td>2</td>
<td>ExactMatch</td>
<td>Iteration²</td>
<td>Iteration²</td>
<td>Delete</td>
<td>Download</td>
</tr>
</tbody>
</table>

Figure 4.8 Experiment Snapshot
the dataset, and the learning type. The other parameters are irrelevant with respect to this operation. Each experiment has its corresponding to each of its associated table accordingly. There are two functionalities to these experiments “download” and “delete”. The download option downloads the details of that associated experiment. So when the download is selected for upload operation, the corresponding data and attribute files associated with the dataset are downloaded to the local disk. These files are referenced and downloaded from “tbRawData”. Since it is more than a single file, the files are zipped and downloaded in zip format. The delete option deletes the reference of an experiment. Deleting an upload experiment is deleting the reference in the “tblRawData”. Hence when the “upload” experiment has been deleted then its reference of its files in the “tblRawData” is also deleted automatically. Moreover if the experiment associated with the upload operation has been deleted then all the experiments associated with the dataset are also deleted, because without the dataset it is meaningless to have the experiments associated with it, and in future also it wouldn’t be possible to perform any experiments on the dataset. Therefore it is necessary to be careful when deleting an experiment.

The next important experiment is “learn” operation. In Chapter II, was shown that learn generates eight different types of files, associating its rule set information. These files and its information are stored in “tblRuleData” table. The experiment associated is shown in Figure 4.8; this operation is called as “learn” referred by expid “3”. The expid 3 has the dataset name on which the rules are generated, and “blem2” is the learning type. The download option would reference the rule data files in the tblRuleData associated with the dataset and are downloaded to the local disk. These files are also zipped before downloaded. The delete option here also deletes the reference of an
experiment i.e., the reference in the tblRuleData table. When the experiment associated
with learn is deleted, the test experiments performed on the learn data are also deleted
because without learn data, test operations cannot be performed.

The next experiment is “generate test file”. Given the split generate test file
generates the test file from the dataset. These test files are saved in “tblTestFile”. In
Figure 4.8, the experiment referred by expid “4” is shown. This experiment is named
“generate test file”, the details associated are the dataset name on which the test file is
generated and the split by which it has been randomly generated from the dataset. So
many test files can be generated on any dataset all are referenced by the above
parameters. If the user has own test file, then the uploaded test file is also referenced by
the same details, except the split here would be zero. Here also there are two operations
“delete” and “download”. The delete operation would delete the reference of an
experiment and also deletes the test file referred by “tblTestFile”. Since the test files are
independent of any operation so deleting the test file wouldn’t delete any other
experiments.

The next experiment is “test” operation. Given the rule data and the test file the
test operation would calculate the accuracy. The details of the test experiment are
referenced by tblTestData table. In Figure 4.8, the experiment “5” refers to the “test”
operation. The test operation is viewed by the dataset name; the test file name by which it
is tested upon, the number of iterations would be obviously one because it is being tested
on single iteration, the matching criteria, the weight calculation method, the learning
type, and the best iteration. The best iteration here is always one since there is only one
iteration that is being tested. So as we can observe, the detail information of the
experiment performed is shown precisely. The download option would download all the files related to this experiment; it would include all the rule data, the test data and the test file used for testing. The delete option would delete the experiment associated and also its reference in the tblTestData, however it doesn’t delete the test file because the test file would be available for the further testing.

The final experiment is “learn and test”. It is the most powerful experiment among all the experiments. The “learn and test” experiments are also referenced by the tblTestData table. In Figure 4.8, the experiment referred by expid “6” is “learn and test” experiment, the operation is called as “learn and test”. The details about the experiments have been discussed in previous Chapters II and III. The “learn and test” details are viewed by:

- Name of the dataset on which the experiment is performed
- The split, used to generate train and test files randomly.
- Number of iterations, implies the number of training and testing files generated.
- The matching criteria either exact match or partial match.
- The weight calculation method, can be certainty * coverage, certainty * strength, certainty alone and coverage alone.
- The learning type, here it is BLEM2.
- The best iterations, depending upon number of iterations the experiment is performed, the maximum accuracy achieved among the iterations is shown in best iteration. As we can see in Figure 4.8, experiment 6 has two best
iterations among 3 iterations; the two iterations, iteration 2 and 3 have achieved same accuracy.

By observing these details, it clearly summarizes the detail of the experiment. The download and delete options are similar to the previous experiments. The download option downloads all the files related to this experiment. All iterations have set of files, so depending upon iteration the files are named appropriately and differentiated. The delete operation deletes the experiment and its reference in the tblTestData table. Since it is an independent experiment deleting the experiment doesn’t affect other experiments.

As we can observe both “test” and “learn and test” experiments are referenced by the same table tblTestData. But deleting the respective experiment doesn’t delete all the experiments associated by the dataset name. It deletes or downloads the experiment that it is referenced by its “id” in the experiment. Each experiment has its id referenced by its table.

The experiment management system helps in understanding the data mining tasks, downloading the respective files, deleting the experiments which are not required. It also extends in giving an intuitive idea and summary information about the experiments held.
CHAPTER V

RUNNING EXAMPLE

The design and implementation of the system have been discussed in previous chapters. This chapter helps us in understanding how the application works in managing the experiments. It explains with a running example of how the system works. The system is being tested on many datasets with different number of attributes and also varying the file size of raw data file. Each and every feature is tested separately with all possibilities, to break the system. The system was stable and worked for all datasets consistently. To understand the system and management let us take a sample dataset, a simple general example of the dataset with the details is given in Figure 5.1 and Figure 5.2. The data set has six attributes and 28 examples.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>a</td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>b</td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>c</td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>e</td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>f</td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>d</td>
<td>c</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.1 Attribute File for Bench Dataset
Initially we need to upload the dataset to the system using upload operation. While uploading the dataset, system prompts for the dataset name, let us name the dataset as “bench”. For further future experiments dataset is referred by bench. Once uploaded the experiment looks like those shown in Figure 5.3. It shows the name as bench, operation as upload and learning type as blem2.

![Table of Data File for Bench Dataset](image)

<table>
<thead>
<tr>
<th>0,0,1,0,0,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1,1,0,0,1</td>
</tr>
<tr>
<td>0,1,0,0,0,2</td>
</tr>
<tr>
<td>1,0,0,0,1,1</td>
</tr>
<tr>
<td>0,1,0,0,0,2</td>
</tr>
<tr>
<td>1,0,0,0,1,3</td>
</tr>
<tr>
<td>0,0,0,1,1,3</td>
</tr>
<tr>
<td>1,1,1,1,1,1</td>
</tr>
<tr>
<td>0,0,0,1,1,3</td>
</tr>
<tr>
<td>0,0,1,0,0,1</td>
</tr>
<tr>
<td>1,1,1,0,0,2</td>
</tr>
<tr>
<td>1,1,1,1,1,2</td>
</tr>
<tr>
<td>1,1,0,1,1,3</td>
</tr>
<tr>
<td>1,1,0,0,1,3</td>
</tr>
<tr>
<td>0,0,1,1,1,1</td>
</tr>
<tr>
<td>1,1,0,1,1,2</td>
</tr>
<tr>
<td>0,0,0,1,1,3</td>
</tr>
<tr>
<td>0,0,0,0,0,2</td>
</tr>
<tr>
<td>0,0,0,0,0,2</td>
</tr>
<tr>
<td>1,1,1,0,0,3</td>
</tr>
<tr>
<td>1,1,0,0,1,2</td>
</tr>
<tr>
<td>0,0,1,0,1,1</td>
</tr>
<tr>
<td>1,1,1,0,0,2</td>
</tr>
<tr>
<td>0,0,1,1,1,2</td>
</tr>
<tr>
<td>1,0,1,0,1,1</td>
</tr>
<tr>
<td>1,0,1,0,1,3</td>
</tr>
<tr>
<td>1,0,1,0,1,2</td>
</tr>
<tr>
<td>1,1,1,1,0,3</td>
</tr>
</tbody>
</table>
The next experiment would be to learn rules from the experiment. By choosing bench as the dataset the rules can be learnt. The derived rules in files are saved accordingly with respect to the bench dataset. After learning the rules the experiment system is shown in Figure 5.4, the dataset name, the operation learn and the learning type is displayed.
Figure 5.4 Experiment Snapshot after Learning

The next experiment is to generate test file. By choosing “bench” dataset, giving split as “20%” the test file is generated; Figure 5.5 shows the details of the experiment. As we can observe the details of the experiment are shown with the split, dataset name, the learning type and the name given to the test file that is generated is shown.
The next experiment is test. The dataset bench is selected; any test file corresponding to this dataset can be selected, the previously generated test file is selected i.e., “bench28test20.txt”, the matching criteria as “exact match” and the weight calculation method as “certainty*coverage” the rules are tested. The details of the experiment are shown below in Figure 5.6, i.e., the dataset name, the operation performed, the test file name on which rules are tested, matching criteria, the weight calculation method are displayed.
The next experiment is learn and test. The experiment is performed with these as input:

- The number of training and testing files to be generated as “3”.
- The split as “20%”.
- The matching criteria as “Exact Match”.
- The weight calculation method as “certainty*coverage”.

The training and testing files are generated, rules are learnt, tested, and the results are saved. The details of the experiment are summarized in Figure 5.7, i.e., the dataset name, the operation performed, the split, the matching criteria, the weight calculation method and the best iterations in this experiment can be viewed.
With all the consolidated experiments on the dataset “bench” the experiments are shown, the files related to the experiments are downloaded and verified that it downloads the correct files. Similarly for different datasets the experiments are performed as shown in Figure 5.8 and Figure 5.9. Figure 5.8 shows first ten experiments, and other ten experiments are shown in Figure 5.9. As we can see with different datasets any operations can be performed and downloaded and can be deleted.
### Experiments

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Event</th>
<th>Tablename</th>
<th>Operation</th>
<th>Testfilename</th>
<th>Split</th>
<th>Nseq</th>
<th>Matching</th>
<th>Weight/Cast Type</th>
<th>LearnType</th>
<th>Restoration</th>
<th>Delete</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/22/2010</td>
<td>24:57</td>
<td>PM</td>
<td>m26</td>
<td>upload</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>24:02</td>
<td>PM</td>
<td>m26</td>
<td>learn</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>24:07</td>
<td>PM</td>
<td>m26</td>
<td>upload</td>
<td>m26xini2000</td>
<td>20</td>
<td>1</td>
<td>NA</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>24:11</td>
<td>PM</td>
<td>m26</td>
<td>test</td>
<td>m26xini2000</td>
<td>0</td>
<td>1</td>
<td>BrainMass</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>24:02</td>
<td>PM</td>
<td>m26</td>
<td>learn</td>
<td>m26xini2000</td>
<td>20</td>
<td>1</td>
<td>BrainMass</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>24:53</td>
<td>PM</td>
<td>m26</td>
<td>upload</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>24:31</td>
<td>PM</td>
<td>m26</td>
<td>learn</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>25:50</td>
<td>PM</td>
<td>m26</td>
<td>upload</td>
<td>m26xini2000</td>
<td>20</td>
<td>1</td>
<td>BrainMass</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>01/22/2010</td>
<td>25:00</td>
<td>PM</td>
<td>m26</td>
<td>test</td>
<td>m26xini2000</td>
<td>0</td>
<td>1</td>
<td>BrainMass</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 5.8 Snapshot for First Ten Experiments
<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Experiment</th>
<th>Status</th>
<th>Module</th>
<th>Test</th>
<th>Match</th>
<th>Certainty</th>
<th>Coverage</th>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/22/2011</td>
<td>bench20</td>
<td>upload</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>v0.2</td>
<td>NA</td>
</tr>
<tr>
<td>2/22/2011</td>
<td>bench20</td>
<td>test</td>
<td>bench3/bench2</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>v0.2</td>
<td>NA</td>
</tr>
<tr>
<td>2/22/2011</td>
<td>bench20</td>
<td>test</td>
<td>bench1/bench2</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>v0.2</td>
<td>NA</td>
</tr>
<tr>
<td>2/22/2011</td>
<td>bench20</td>
<td>test</td>
<td>bench1/bench2</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>v0.2</td>
<td>NA</td>
</tr>
</tbody>
</table>

Figure 5.9 Snapshot for Next Ten Experiments
CHAPTER VI

DISCUSSIONS AND FUTURE WORK

This chapter summarizes the main contributions and conclusions of this thesis regarding the management of files in the machine learning tool. Moreover, this chapter also addresses some future work based on the current work.

6.1 Contributions and Evaluations

The Rule-based machine learning utility can be operated upon a single dataset at a time. The experiments once performed cannot be viewed or retrieved in the future. Each time we wish to perform an experiment on different dataset the dataset has to be uploaded again. It would be difficult for the user to perform similar experiments more than once and upload similar datasets each time when the data mining task to be performed. To overcome these limitations our research has presented an experiment management system.

The main purpose of this thesis was to develop an experiment management system for managing data mining tasks. This system helps the user to manage the experiments being performed. It provides an easy way to manage the datasets and provides consolidated view of experiments, with additional functionalities with respect to each experiment. The system has been evaluated with many different datasets with different sizes. The system is stable and consistent for all the types of datasets.
The system shows the summary of the results, wherein the user can compare the results and gives an idea of the experiments to be performed for the desired results. It also gives an idea of the parameters need to be changed for the better results and accuracy.

In our research, we demonstrated the features of the Rule-based Machine Learning utility, discussed the limitations and proposed a new system to overcome these limitations. The system has been designed and implemented successfully.

6.2 Future Work

This research developed a system for managing experiments in data mining tasks. It also demonstrated the functionalities of the management of experiments in the Rule-based Machine Learning Utility. However the management system has been developed for only BLEM2 learning type, the system can be extended and supported for other learning types.

During the management, the system writes some files to the local disk and then saves to the database. For example when the dataset has to be learned, the learned files are written to the local disk and then saved to the database. The system can be developed such that the files are written directly to the database. The system can be improved to be more robust and give better performance when large sizes of datasets are uploaded. The implementation can also be extended by confirming its standards with the work flow management [21, 24]. These are some of the future works that can be addressed.
REFERENCES


[3] Zdravko Markov, Ingrid Russell. An Introduction to the WEKA Data Mining System


APPENDICES
APPENDIX A

SOURCE CODE WRITING FILES TO DATABASE

This section consists of the source code for writing file to the table, which is written in the C# programming language.

```csharp
public int WriteToDB1(string filename, string tablename, string strName, string strType, string learningtype, ref byte[] Buffer, string todaydate, string delimiter)
{
    int nFileID = 1;
    //string temp;
    //byte[] tempbyte;

    //Get the new fileID
    // Create connection
    SqlConnection dbConn = new SqlConnection(ConfigurationSettings.AppSettings["ConnectionString3"]);

    string id = null;
    if (tablename == "tblRawData")
        id = "rawdataid";
    else if (tablename == "tblRuleData")
        id = "rulesetid";

    SqlDataAdapter dbAdapt2 = new SqlDataAdapter("SELECT MAX(" + id + ") FROM " + tablename, dbConn);
    // New DataSet2
    DataSet dbSet2 = new DataSet();
    dbAdapt2.Fill(dbSet2, tablename);
    DataTable dbTable2 = dbSet2.Tables[tablename];
    // Create new row
    DataRow dbRow2 = dbTable2.Rows[0];
    if (!dbRow2.IsNull(0))
        nFileID = (int)dbRow2[0] + 1;
```
// Create Adapter
SqlDataAdapter dbAdapt = new SqlDataAdapter("SELECT * FROM " + tablename, dbConn);
// We need this to get an ID back from the database
dbAdapt.MissingSchemaAction = MissingSchemaAction.AddWithKey;

// Create and initialize CommandBuilder
SqlCommandBuilder dbCB = new SqlCommandBuilder(dbAdapt);

// Open Connection
dbConn.Open();

// New DataSet
DataSet dbSet = new DataSet();

// Populate DataSet with data
dbAdapt.Fill(dbSet, tablename);

// Get reference to our table
DataTable dbTable = dbSet.Tables[tablename];

// Create new row
DataRow dbRow = dbTable.NewRow();

// Store data in the row
dbRow["tablename"] = filename;
dbRow["filename"] = strName;
dbRow["filesize"] = Buffer.Length;
dbRow["filedata"] = Buffer;
dbRow["learningtype"] = learningtype;
dbRow["date"] = todaydate;
if (tablename == "tblRawData")
{
    dbRow["delimiter"] = delimiter;
    dbRow["rawdataid"] = nFileID;
}
else if (tablename == "tblRuleData")
    dbRow["rulesetid"] = nFileID;

// Add row back to table
dbTable.Rows.Add(dbRow);

// Update data source
dbAdapt.Update(dbSet, tablename);

// Get newFileID
if (tablename == "tblRawData")
{
    if (!dbRow.IsNull("rawdataid"))
        nFileID = (int)dbRow["rawdataid"];             
}
else if (tablename == "tblRuleData")
{

if (!dbRow.IsNull("rulesetid"))
    nFileID = (int)dbRow["rulesetid"];}
//tempbyte=(byte [])dbRow["filedata"];
//temp=System.Text.Encoding.ASCII.GetString(tempbyte,0,tempbyte.Length);
// Close connection
dbConn.Close();
// Return FileID
return nFileID;
}
This section consists of the source code for writing the experiment information at the end of each data mining task. This is written in the C# programming language.

```csharp
public int WriteToDBExperimentData(string filename, string tablename,int fileid,string operation, string testFilename, int split, int iterations, string MatchingCriteria, string weighcalc, string learningtype, string date,string bestiteration)
{
    int nFileID = 1;
    string temp;
    byte[] tempbyte;

    //Get the new fileID
    // Create connection
    SqlConnection dbConn = new
    SqlConnection(ConfigurationSettings.AppSettings["ConnectionString3"]) ;

    SqlDataAdapter dbAdapt2 = new SqlDataAdapter("SELECT MAX(experimentid) FROM "+ tablename, dbConn);
    // New DataSet2
    DataSet dbSet2 = new DataSet();

    // Populate DataSet2 with data
    dbAdapt2.Fill(dbSet2, tablename);
    // Get reference to our table
    DataTable dbTable2 = dbSet2.Tables[tablename];

    // Create new row
    DataRow dbRow2 = dbTable2.Rows[0];
    if (!dbRow2.IsNull(0))
        nFileID = (int)dbRow2[0] + 1;

    // Create Adapter
    SqlDataAdapter dbAdapt = new SqlDataAdapter("SELECT * FROM " + tablename, dbConn);
```
// We need this to get an ID back from the database
dbAdapt.MissingSchemaAction = MissingSchemaAction.AddWithKey;

// Create and initialize CommandBuilder
SqlCommandBuilder dbCB = new SqlCommandBuilder(dbAdapt);

// Open Connection
dbConn.Open();

// New DataSet
DataSet dbSet = new DataSet();

// Populate DataSet with data
dbAdapt.Fill(dbSet, tablename);

// Get reference to our table
DataTable dbTable = dbSet.Tables[tablename];

// Create new row
DataRow dbRow = dbTable.NewRow();

// Store data in the row
dbRow["tablename"] = filename;
dbRow["referenceid"] = fileid;
dbRow["operation"] = operation;
dbRow["testfilename"] = testFilename;
dbRow["split"] = split;
dbRow["noofiterations"] = iterations;

dbRow["MatchingCriteria"] = MatchingCriteria;
dbRow["WeightCalculation"] = weighcalc;
dbRow["learningtype"] = learningtype;

dbRow["bestiteration"] = bestiteration;
dbRow["date"] = date;
dbRow["experimentid"] = nFileID;

// Add row back to table
dbTable.Rows.Add(dbRow);

// Update data source
dbAdapt.Update(dbSet, tablename);

// Get newFileID
if (!dbRow.IsNull("experimentid"))
    nFileID = (int)dbRow["experimentid"];  
//tempbyte=(byte [])dbRow["filedata"];
//temp=System.Text.Encoding.ASCII.GetString(tempbyte,0,tempbyte.Length);
// Close connection
dbConn.Close();
// Return FileID
return nFileID;
}