A TOOL FOR PUBLIC ANALYSIS OF SCIENTIFIC DATA

D Haglin[*1], *R Roiger*[2], *J Hakkila*[3] *and T Giblin*[4]

[*1] *Computer and Information Sciences Dept., Minnesota State University, Mankato, USA.*
EMail: David.Haglin@mnsu.edu
[2] *Computer and Information Sciences Dept., Minnesota State University, Mankato, USA.*
EMail: Richard.Roiger@mnsu.edu
[3] *Department of Physics and Astronomy, The College of Charleston, South Carolina, USA.*
EMail: hakkilaj@cofc.edu
[4] *Department of Physics and Astronomy, The College of Charleston, South Carolina, USA.*
EMail: giblint@cofc.edu

ABSTRACT

The scientific method encourages sharing data with other researchers to independently verify conclusions. Currently, technical barriers impede such public scrutiny. A strategy for offering scientific data for public analysis is described. With this strategy, effectively no requirements of software installation (other than a web browser) or data manipulation are imposed on other researchers to prepare for perusing the scientific data. A prototype showcasing this strategy is described.

Keywords: Scientific Data, Data Mining, Machine Learning, KDD, Gamma Ray Burst.

1 INTRODUCTION

Scientific data is often gathered with the intention of conducting a thorough analysis. Unfortunately, gathered data are frequently archived and eventually forgotten. This results in lost potential for increasing scientific returns. Software tools have the potential to mitigate this loss, but existing tools suffer from several deficiencies.

1.1 Problems with current software

In this paper we address two problems with using existing software tools for the purpose of data analysis. The first problem occurs when multiple tools are used for data analysis. The second problem is the data preparation imposed on the scientist.

1.1.1 Using multiple tools

Many commercial and non-commercial data analysis tools are currently available. Descriptions and/or links to many of these tools can be obtained by visiting `www.kdnuggets.com`. Some of the more popular commercial products include *Clementine*, *DBMiner*, *IDL* (the Interactive Data Language), *Polyanalyst*, *SAS* and *SPSS*. Among the commonly used non-commercial products are *C4.5*, *Gnome Data Mining Tools*, *IBM Intelligent Miner* and *Weka*.

The use of software tools to aid scientific data analysis faces many obstacles. As each data analysis tool has its own set of strengths and weaknesses, using multiple tools can provide a more thorough scientific investigation. We therefore first consider the obstacle of incompatibilities among the various tools.

Scientific analysis is often performed with non-commercial products. Unfortunately, non-commercial tools are particularly riddled with inconsistencies. Most of the non-commercial tools have command-line interfaces or non-intuitive GUI's and require manual manipulation of several files. Data formatting requirements vary greatly among available tools. This often requires the user to perform several tedious data manipulation operations in order to obtain a complete analysis of the data.

We note that efforts have been proposed to standardize the internal representation of data (Grossman, Hornick, & Meyer, 2002). Such a standard would reduce the data format transformation required to use multiple tools. Although a step in the right direction, the plethora of existing tools would need to be reengineered to accommodate such a standard.

1.1.2 Data preparation

As we worked with various data analysis software a second obstacle emerged. The available tools do not sufficiently aid the scientist in gathering and preparing the science data in a form most likely to produce valuable analysis. We invoke a phrase from early computing — *garbage in, garbage out* — to describe the importance of the initial data preparation. Smyth, Pregibon, & Faloutsos (2002) points out that an area conspicuously absent in data mining research, yet tremendously important in practically any scientific context, is human-computer interaction (HCI). Our experience suggests that significant HCI is necessary during data preparation.

1.2 Our proposed solution

To facilitate public analysis of scientific data it is necessary to provide the research community with: (i) widespread access to useful data, (ii) easy-to-use data selection and preprocessing capabilities, and (iii) data analysis software. A web-based GUI interface to a database engine providing data selection capabilities may be used to satisfy criteria (i) and (ii), assuming that useful data has been gathered and prepared. From a web browser the user may select parts of the data relevant to their line of inquiry.

Several categories of data analysis software (criteria iii) exist: data visualization tools, *machine learning* implementations, and statistical packages. We define machine learning as the process of forming general concept definitions by observing specific examples of concepts to be learned.

To assist in the public analysis of scientific data we propose to integrate a GUI database interface with data analysis software as part of a single Web application. Note that this strategy addresses only the tool incompatibility issue and leaves the data preparation issues unresolved. To demonstrate the efficacy of our strategy we have developed a prototype application to aid Gamma-Ray Burst (GRB) astronomers that we call the GRB ToolShed (Haglin, Roiger, Hakkila, Pendleton, & Mallozzi, 2000; Hakkila, Haglin, Roiger, Giblin, Paciesas, & Meegan, 2001) available at `http://grb.mnsu.edu/`.

Our solution removes tool inconsistency problems (the first obstacle) but only partially addresses the many issues surrounding data preparation (the second obstacle). The rest of this paper is organized as follows. Section 2 describes a common view of data analysis giving particular attention to the interplay between the Scientific Method and data mining. Section 3 provides a design of a general application "ToolShed" solution. Section 4 presents our prototype: the GRB ToolShed. Ideas for further work are presented in section 5.

2 SCIENTIFIC DATA ANALYSIS

To a scientist, the overarching goal of data analysis (from the scientific method) is to model nature's behavior and to use that knowledge to predict future natural behaviors. The scientific method — along with a discussion of the interplay between the scientific method and data mining — is described below.

The scientific method

A. A natural phenomenon is observed or an experimental result is obtained (preferably by several independent experimenters). The accuracy of the observations is described by experimental uncertainties.

B. An hypothesis is formulated to explain the phenomenon. The hypothesis can take the form of a causal mechanism or an empirical one, and is preferably modeled mathematically. The mathematical model must satisfy the observations within experimental uncertainties. The model must also be consistent with other related observations.

C. The model is used to predict the existence of other phenomena, or to predict quantitatively the results of new observations. The process repeats at step A in order to continually improve the understanding of natural phenomena.

Modern application of the Scientific Method has been critically aided by significant improvements in computational capacity. Computation is used to simplify data collection and to assist scientists with complex mathematical modeling. It also improves mathematical modeling in two very different ways: it is used to identify patterns and empirical relationships in the data, and it is used in the development of physical models.

Data mining is a general computational technique used to describe an efficient method for identifying patterns and empirical data relationships. We use terminology consistent with Fayyad, Piatetsky-Shapiro, & Smyth, (1996a) where data mining is one step of a larger strategy know as knowledge discovery in databases (the KDD process). The KDD process can be viewed as applying the scientific method to discovering patterns in data. It can therefore be used (recursively) as part of step B of the Scientific Method when patterns and empirical relationships in the data are needed.

Application of the KDD process to scientific data is not new. Weir, Fayyad, & Djorgovski, (1995) describe a KDD tool to successfully classify deep-sky objects represented by 40 features. Only eight features were found necessary to increase the number of correctly classified objects by 300%. Other successful applications of KDD scientific classification are summarized in Langley & Simon, (1995) and Fayyad, Haussler, & Stolorz, (1996a). An appeal for more research on "highly automated, scalable, integrated, reliable data mining systems and tools" appears in a discussion of emerging scientific applications in data mining (Han, Altman, Kumar, Mannila, & Pregibon, 2002).

2.1 A KDD process model

Figure 1 offers a schematic diagram showing our KDD process model. The flows in the figure show the most common paths suggested by our model, although other paths are possible as we describe later. Note that the Scientific Evaluation step typically involves human intervention.

A common misconception about KDD (in particular, the data mining step) is that it is somehow capable of replacing a scientist. That might happen if KDD were a process that simply looked at data (as the only input) and produced knowledge (as output) without any guidance. But KDD is really only a very useful tool to aid scientists. *It is up to the scientist to formulate goals, present data appropriate to those goals, and draw scientific conclusions from the patterns identified by the data mining step.*

Details about each step of our process model, as seen in Figure 1, are given below:

1. **Goal identification**. The KDD process cannot begin without a goal. The focus is on understanding the scientific domain under consideration. The scientist writes a clear statement about what is to be accomplished. An hypothesis offering a likely or desired outcome can also be stated.

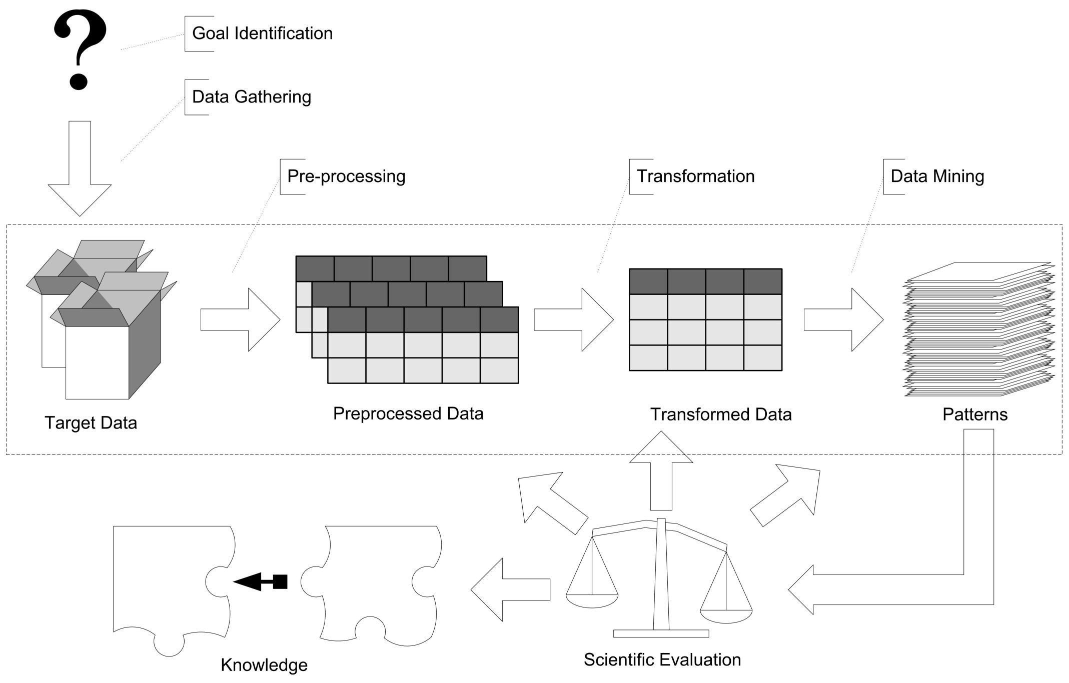


Figure 1. A KDD Process Model

2. **Target data creation**. With the help of one or more scientists and knowledge discovery tools, an initial set of data to be analyzed is prepared.
3. **Data preprocessing**. Noisy data can be a problem for many data analysis tools. The scientist must account for noisy data such as incorrect feature values, missing values and feature values with large measurement errors.
4. **Data transformation**. Features and instances are added and/or eliminated from the target data. Methods to normalize, convert and smooth data must be determined.
5. **Data mining**. A best model for representing the data is created by applying one or more data mining algorithms.
6. **Evaluation and interpretation**. A scientist examines the output from step 5 to determine if what has been discovered is both useful and interesting. Decisions are made about whether to repeat previous steps using new features and/or instances.
7. **Take action**. If the discovered knowledge is deemed useful, the knowledge is incorporated and applied directly to appropriate problems.

We emphasize the intrinsically iterative nature of this process model. Although not explicit in the diagram, it is assumed that from any of the steps it may be necessary to return to any previous step. For example, if scientific evaluation concludes that interesting patterns have not been uncovered, a better outcome may be produced by going back to the data transformation step to select a different set of features and/or instances.

2.2 Data mining: A closer look

Although research has focused on all aspects of KDD, the main thrust has been on data mining (the KDD step that applies algorithms to extract patterns from data). These patterns represent concept classes identified in the data set. Any one of three general data mining strategies can be applied: *supervised learning*, *unsupervised clustering*, and *affinity analysis*.

2.2.1 Data mining strategies

Supervised learning is a two-phase process where one feature is identified as the dependent variable (output feature) and the remaining features are assumed to be independent variables (input features). During the *training phase*, the set of input features are used to construct a generalized model of the data whose purpose is to determine the value of the output feature. The *test phase* uses the constructed model to evaluate *unseen instances*. Supervised strategies can be subdivided according to whether the evaluation targets current conditions, such as the presence of an atmosphere on an astronomical body, or future outcomes, such as the likelihood of an earthquake in California.

With unsupervised clustering we are without a dependent variable to guide the learning process. Rather, the learning algorithm builds a generalization of the data by using a defined measure of cluster quality to group instances into two or more clusters. Unsupervised clustering is often used to determine if meaningful relationships can be found in the data. It can also be used to determine a best set of input features for supervised learning, to evaluate the likely performance of a supervised learning model and to detect atypical instances known as outliers with the data.

Affinity analysis aims to determine those attributes that "belong together." The relationships are typically described using *association rules*, which take the form: `if antecedent then consequent`. This strategy is appropriate for all types of datasets. The approach is best used when we are interested in finding multiple relationships among a set of features. Note that a feature appearing in the antecedent of one rule may appear in the consequent of another rule. Affinity analysis is often used to help retailers arrange shelf or catalog items, design promotions, and develop cross-marketing strategies.

2.2.2 Data mining techniques

A data mining technique — defined by an algorithm and an associated data structure such as a tree or a set of rules — is used to apply a data mining strategy to a set of data. Here we detail two common data mining techniques: one for supervised learning and the other for unsupervised clustering.

Decision trees (Quinlan, 1986; Quinlan, 1993) are the most studied of all supervised techniques. A decision tree is a simple structure where non-terminal nodes represent tests on one or more attributes and terminal nodes reflect decision outcomes. Decision trees have several advantages in that they are easy for humans to understand, can be transformed into rules and have been shown to work well in practice.

Consider the hypothetical data set given in table 1 (Roiger & Geatz, 2003). Figure 2 shows a decision tree representing a generalization of the data in table 1.

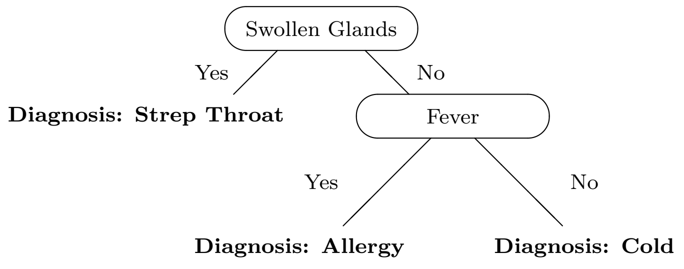


Figure 2. Decision Tree Method of Classification

Table 1. Hypothetical Training Data for Disease Diagnosis

Id#	**Sore Throat**	**Fever**	**Swollen Glands**	**Congestion**	**Headache**	**Diagnosis**
1	Yes	Yes	Yes	Yes	Yes	Strep Throat
2	No	No	No	Yes	Yes	Allergy
3	Yes	Yes	No	Yes	No	Cold
4	Yes	No	Yes	No	No	Strep Throat
5	No	Yes	No	Yes	No	Cold
6	No	No	No	Yes	No	Allergy
7	No	No	Yes	No	No	Strep Throat
8	Yes	No	No	Yes	Yes	Allergy
9	No	Yes	No	Yes	Yes	Cold
10	Yes	Yes	No	Yes	Yes	Cold

To translate directly to a set of *rules*:

- If a patient has swollen glands, the disease diagnosis is “strep throat.”
- If a patient does not have swollen glands and has a fever, the diagnosis is a “cold.”
- If a patient does not have swollen glands and does not have a fever, the diagnosis is an “allergy.”

The decision tree tells us that we can accurately diagnose a patient in this data set by concerning ourselves only with whether the patient has swollen glands and a fever. The features *sore throat*, *congestion* and *headache* do not play a role in determining a diagnosis. As we can see, the decision tree has generalized the data and provided us with a summary of those features and feature relationships important for an accurate diagnosis.

The instances used to create the decision tree model are known as *training data*. At this point, the training instances are the only instances known to be correctly classified by the model. However, our model is useful to the extent that it can correctly classify new instances of unknown origin. Therefore, to determine how well the model is able to be of general use, we test the accuracy of the model using a *test data set*. The instances of the test set have a known classification. Therefore, we can compare the test set instance classifications determined by the model with the correct classification values. Test set classification correctness gives us some indication about the future performance of the model.

Decision trees are typically constructed using only those features best able to differentiate the concepts to be learned. Feature selection is often accomplished, using a formula from information theory, by choosing the feature with the smallest average disorder as defined below.

$$\text{Average disorder} = \sum_b \left(\frac{n_b}{n_t}\right) \times \left(\sum_c -\frac{n_{bc}}{n_b} \log_2 \frac{n_{bc}}{n_b}\right),$$

where:

n_b	is the number of samples in branch b,
n_t	is the total number of samples in all branches,
n_{bc}	is the total of samples in branch b of class c.

Other common supervised techniques include backpropagation neural networks (Jain, Mao, & Mohiuddin, 1996), linear regression analysis, logistic regression, and Bayes classifier.

The K-Means algorithm (Lloyd, 1982) is a simple yet effective unsupervised clustering technique. The first step of the algorithm requires an initial decision about how many clusters are present in the data. Next, the algorithm randomly selects K data points as initial cluster centers. Each instance is then placed in the cluster to which it is most similar. Once all instances have been placed in their appropriate cluster, the cluster centers are updated by computing the mean of each new cluster. The process continues until an iteration of the algorithm shows no change in the cluster centers.

The K-Means algorithm is guaranteed to cluster the data instances into a stable state. However, the clustering may not be globally optimal. The algorithm works best when the clusters that exist in the data are of approximately equal size. Several applications of the algorithm to the data may be necessary to achieve an acceptable result. Other unsupervised clustering techniques include conceptual clustering (Gennari, Langely, & Fisher, 1989), agglomerative clustering, and expectation maximization (Dempster, Laird & Rubin, 1977).

2.3 Evaluation methods

Scientific evaluation is probably the most critical of all the steps in the KDD process model. The purpose of a scientific evaluation is to determine whether a learner model is acceptable and can be applied to problems outside the realm of a test environment. If acceptable results are achieved, it is at this stage where acquired knowledge is translated into terms appropriate to the science discipline.

Scientific evaluation can take many forms, some of which include (Roiger & Geatz, 2003):

Statistical analysis. Such analysis is useful for determining if significant differences exist between the performance of various data mining models created using distinct sets of attributes and instances.

Heuristic analysis. Heuristics are "rules of thumb" that generally give solutions to problems where the solution quality is within some acceptable tolerance. Most data mining tools offer numerically computed heuristics to help us decide the degree to which discovered knowledge is of value. One example is the sum of squared error computation associated with the K-Means algorithm.

Experimental analysis. Some data mining techniques build slightly different models each time they are invoked with the same set of parameters and the same data. Other methods build distinct models with slight variations in data selection or parameter settings. Because of this, experimenting with various attribute or instance choices as well as alternative parameter settings can give markedly different results.

Human analysis. The human component of result analysis reminds us that we are in control of the experimental process. In the final analysis, we must decide whether the knowledge gained from a data mining process can be successfully applied to new problems.

Figure 3 shows the major components used to create and test a supervised learner model. All elements contribute in some way to the performance of a created model. When a model fails to perform as expected, an appropriate strategy is to evaluate the affect every component has on model performance. The following is a list of the components shown in the figure together with additional considerations for evaluation.

Features. Feature evaluation focuses on how well the attributes define the domain instances. If two candidate input attributes are highly correlated in a positive or a negative direction, only one of the attributes should be selected for data mining. Statistical tests of significance can be used for numerical attribute evaluation (Weiss, & Indurkhya, 1998).

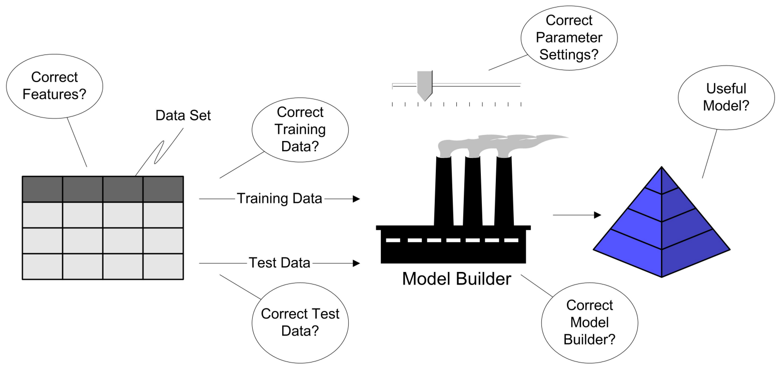


Figure 3. Evaluation of supervised learning.

Training data. If a supervised learner model shows a poor test set accuracy, part of the problem may lie with the training data. Models built with training data that do not represent the set of all possible domain instances or contains an abundance of atypical instances are not likely to perform well. A best preventative measure is to randomly select training data, making sure the classes contained in the training data are distributed as they are seen in the general population.

Test data. The purpose of test data is to offer a measure of future model performance. Test data should be selected randomly.

Parameter settings. Most data mining models support one or more user-specified learning parameters. The technique used to compare supervised learner models built with different data mining techniques can also be applied to compare models constructed with the same data mining method but alternate settings for one or more learning parameters.

Model builder. It has been shown that supervised learner models built with alternative learning techniques tend to show comparable test set error rates. However, there may be situations where one learning technique is preferred over another. For example, neural networks tend to outperform other supervised learning techniques when the training data contains a wealth of missing or noisy data items. Once again, a statistical analysis can be used to decide if two supervised learner models built with the same training data show a significant difference in test set performance (Weiss, & Indurkhya, 1998).

Useful model. Supervised learner models are usually evaluated on test data. Special attention may be paid to the cost of different types of misclassification. For example, we might be willing to use a loan application model that rejects borderline individuals who would likely pay off a loan provided the model does not accept strong candidates for loan default. Test set confidence intervals can be computed for supervised models having either categorical or numeric output (Weiss, & Indurkhya, 1998).

Figure 3 also applies to unsupervised clustering, with the exception that we are without test data containing instances of known classification.

3 OUR ToolShed STRATEGY

Figure 4 depicts a high-level view of the architecture for our strategy. We call this strategy a *ToolShed*, (derived from a **SH**ell for **E**xplorations using **D**ata mining). The basic idea is to allow a scientist at any location on the internet to run a browser aimed at the ToolShed server. The desired outcome is to present the scientist with information that can lead to new understanding of the scientific phenomenon represented or captured in the data.

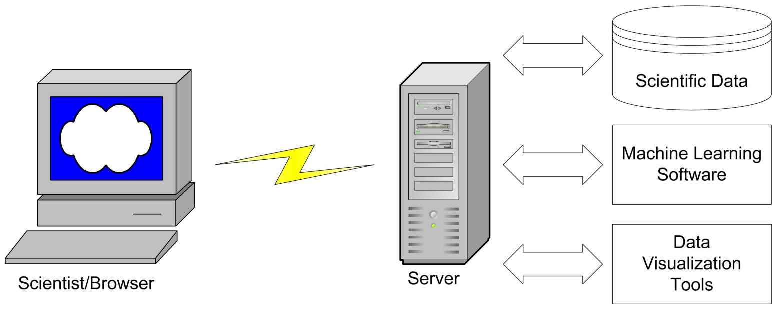


Figure 4. High-level architecture.

The server in figure 4 is the central point of control. It is responsible for presenting information to and requesting information from the user. It also interacts with data analysis software and communicates with the data repository. Classic examples of data repositories include relational database systems such as MySql (`www.mysql.com`) and Postgresql (`www.postrgresql.org`). However, other repository systems such as XML databases may be used.

Once a subset of data has been chosen by the user, the ToolShed application passes the data to either a machine learning algorithm, a statistical tool or a visualization system. All of these "handoffs" are guided by the user. For example, if a ToolShed has multiple machine learning modules, the user is first asked to select a machine learning algorithm to invoke. The user then supplies values for tunable parameters specific to the selected algorithm. Results of the data analysis are sent to the client browser for display.

One feature of our ToolShed design that is essential to scientific inquiry is to allow the user/scientist to include their own data for purposes of data analysis. Another task required of a ToolShed is to support simple computations for data transformation. An example of this is the transformation of the values of a feature into their corresponding logarithmic values.

Standardization of our ToolShed design is desirable. To accomplish this we can leverage some industry standards — Java Server Pages (JSP) and JDBC for access to a relational database — and use freely available data analysis tools. Examples of free tools include: Gnuplot for data visualization (`www.gnuplot.info`); collections of data mining implementations such as Weka (`www.cs.waikato.ac.nz/ml/weka/`) and Orange (`www.ailab.si/orange`); and JSP implementations called Resin (`www.caucho.com`) and Tomcat (`jakarta.apache.org/tomcat/`).

4 OUR PROTOTYPE: THE GRB ToolShed

To demonstrate the efficacy of our strategy we have developed a prototype application to aid Gamma-Ray Burst (GRB) astronomers. Our prototype is designed to meet the objectives of widespread access and ease-of-use, and contains several data analysis tools (Haglin et al., 2000; Hakkila et al.,2001).

4.1 Gamma-ray bursts

Gamma-Ray Bursts are brief flashed of gamma rays whose origins are extragalactic. Astronomers continue to debate the physical phenomenon responsible for creating these bursts. The energy released in a burst is second only to the energy released at the origin of the universe.

The Earth's atmosphere shields its surface from bombardment of gamma ray bursts. This is necessary for survival of life on Earth, but it does mean that collecting data about these GRB events must be done by an observatory above the atmosphere; increasing the costs associated with measuring GRBs.

The data we use was collected by the Burst And Transient Source Experiment (BATSE) aboard NASA's Compton Gamma-Ray Observatory (CGRO) (Paciesas, Meegan, Pendleton, Briggs, Kouveliotou, Koshut at al., 1999). It represents the largest gamma-ray burst database in existence collected by a single instrument. Single-instrument observations are advantageous because systematic effects can be potentially understood and either be removed or corrected, whereas it is much more difficult to disentangle statistical and systematic uncertainties in databases where observations have been collected by multiple instruments. There are 45 features associated with each burst instance. We currently have data on 2702 bursts collected between 1991 and 2000. Note that the CGRO instrument was deactivated in 2000. While this is considered a small data set for data mining, the cost of acquiring additional data is very high and more traditional analysis strategies have produced only a modest understanding of the science.

We look forward to the opportunity to augment our data with new observations recorded by the Swift satellite (`swift.gsfc.nasa.gov`) which was launched on November 20, 2004. This new instrument will record burst features not previously measured by the BATSE instrument.

4.2 GRB ToolShed features

The GRB ToolShed is a reference prototype of our general ToolShed strategy. The features of the GRB ToolShed evolved to its current implementation guided in part by the GRB research community. This guidance resulted in new research contributions (Hakkila, Haglin, Pendleton, Mallozzi, Meegan, & Roiger, 2000; Hakkila, Giblin, Roiger, Haglin, Paciesas, & Meegan, 2003). Most of the general ToolShed features described in section 3 are implemented in our prototype.

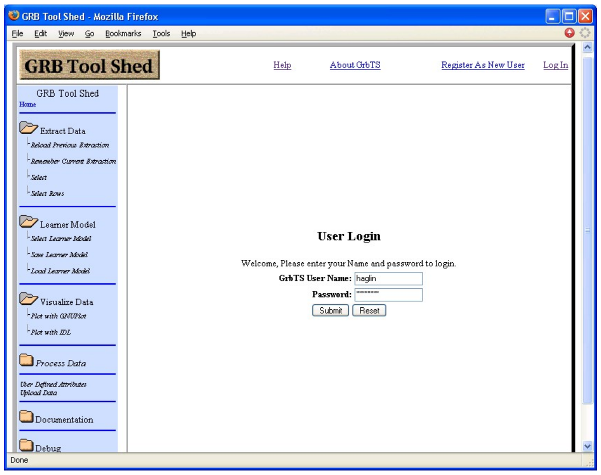
The data mining techniques currently available in the prototype include both supervised learning and unsupervised clustering methods. For supervised learning, there is a java-based version of C4.5 (Quinlan, 1993) called J48, M5′, Linear Regression, K*, Naïve Bayes, support vector machines (SVM), and a Back-Propagation Neural Network. For unsupervised clustering, Estimation Maximization (EM), KMeans, Cobweb and a Kohonen Neural Network are available. See Hand, Mannila, & Smyth, (2001) for background information on these data mining algorithms.

The data visualization tools currently supported by the prototype include both free and commercial software. `Gnuplot`, a free data graphing tool, and a commercial product commonly used by Astronomers called IDL (`www.rsinc.com`) are available.

4.3 The GRB ToolShed in action: supervised learning

Here we demonstrate our prototype by performing a supervised learning session using a variant of the BATSE data. We augmented the BATSE data by attaching an output feature representing a class designation. Class membership was obtained from an unsupervised mining session resulting in three clusters of GRBs (Mukherjee, Feigelson, Babu, Murtagh, Fraley, & Raftery, 1998). The three clusters were labelled *long*, *intermediate*, and *short*. The astronomical implications of our investigation are outside the scope of this paper (Hakkila et al., 2003).

We present a sequence of screen shots to help the reader visualize a complete mining session. The first screen (figure 5) shows the layout of the GRB ToolShed. As can be seen in the figure, a navigation menu appears along the left side of the screen. As a potentially large amount of work is necessary to set up a single data mining session the ToolShed provides capabilities of storing "setup" information on a per user basis. Examples include selecting a subset of features and/or rows (instances) of the data and defining new data that is directly computable from existing data. Previous setup information is tracked on a per user basis by requiring users to log in to the ToolShed.



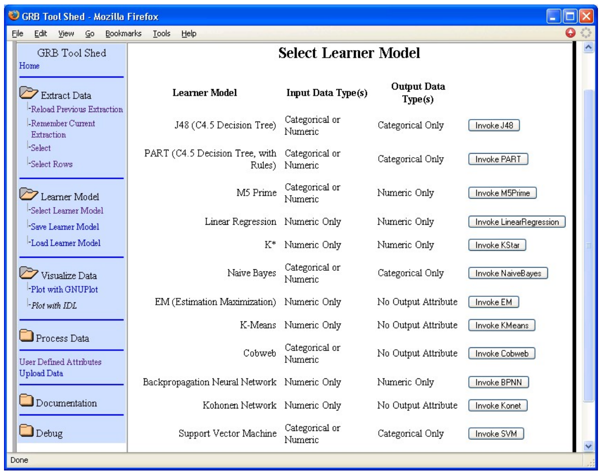
User Login

Welcome, Please enter your Name and password to login.

GrbTS User Name: haglin

Password: ********

Figure 5. Logging in.



Select Learner Model

Learner Model	Input Data Type(s)	Output Data Type(s)	
J48 (C4.5 Decision Tree)	Categorical or Numeric	Categorical Only	Invoke J48
PART (C4.5 Decision Tree, with Rules)	Categorical or Numeric	Categorical Only	Invoke PART
M5 Prime	Categorical or Numeric	Numeric Only	Invoke M5Prime
Linear Regression	Numeric Only	Numeric Only	Invoke LinearRegression
K*	Numeric Only	Numeric Only	Invoke KStar
Naive Bayes	Categorical or Numeric	Categorical Only	Invoke NaiveBayes
EM (Estimation Maximization)	Numeric Only	No Output Attribute	Invoke EM
K-Means	Numeric Only	No Output Attribute	Invoke KMeans
Cobweb	Categorical or Numeric	No Output Attribute	Invoke Cobweb
Backpropagation Neural Network	Numeric Only	Numeric Only	Invoke BPNN
Kohonen Network	Numeric Only	No Output Attribute	Invoke Konet
Support Vector Machine	Categorical or Numeric	Categorical Only	Invoke SVM

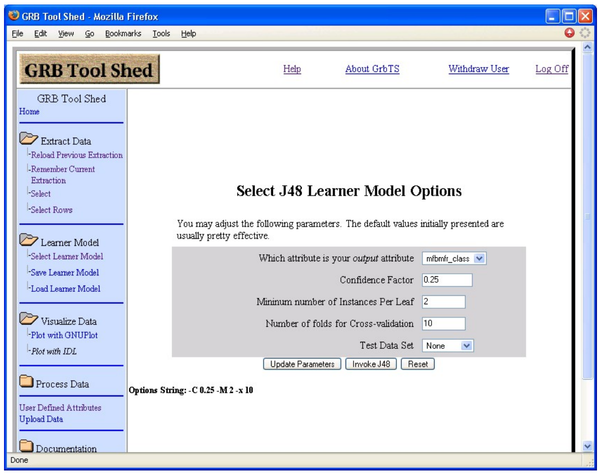
Figure 6. Selecting a learner model.

Once logged in, the user navigates to perform various functions. One typical function allows the user to select a subset of the scientific data in the ToolShed. Our prototype contains several catalogs of Gamma-Ray Burst Astronomy data. Note that the user can have the web application save a selection of data for later retrieval. This feature is indicated in the navigation menu as *remember current extraction* and *reload previous extraction*.

Upon selecting the data the user may initiate the data mining step of the KDD process by choosing one of several learner models. Figure 6 shows a list of the available learner models in our prototype, along with any restrictions each model imposes on the input and output features. Our illustration uses the J48 learner model.

Once a model is selected, additional information specific to that model is requested. For our example, after invoking the decision-tree learner the user is given the option to modify tuning parameter values specific to J48 (see figure 7). The initial values displayed for the tunable parameters are considered reasonably effective default values. The user should change the parameter values only if the learning algorithm is not providing a satisfactory result.

As our prototype uses the WEKA software to mine the data, we are limited to text-based output. Recall that J48 is a decision tree learner model whose output is best displayed as a tree structure. Therefore, we added a Java applet to parse the WEKA output and construct a graphical view of the decision tree, shown in figure 8.



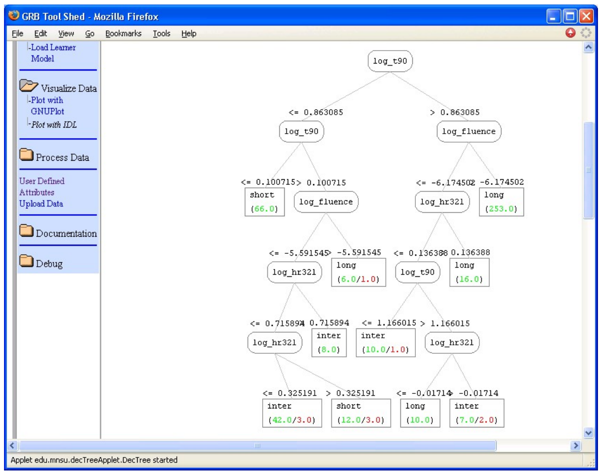
Select J48 Learner Model Options

You may adjust the following parameters. The default values initially presented are usually pretty effective.

Which attribute is your *output* attribute	mfbmfr_class
Confidence Factor	0.25
Mininum number of Instances Per Leaf	2
Number of folds for Cross-validation	10
Test Data Set	None

Options String: -C 0.25 -M 2 -x 10

Figure 7. Tunable parameters for J48.



```
Applet edu.mnsu.decTreeApplet.DecTree started
```

Figure 8. A decision tree.

Although the tree shown in figure 8 is complex it does provide insight into variations between the classes. As examples, the following are "rules" that are taken directly from the decision tree:

```
log_t90 > 0.863085 AND log_fluence > -6.174502:  long
log_t90 <= 0.100715:  short
```

4.4 The GRB ToolShed in action: unsupervised clustering

We modified the data set used in the supervised classification example by removing the "class" feature. This modified data set was then presented to the KMeans machine learning algorithm. One of the tunable parameters for the KMeans algorithm is the number of classes to search for within the data. We asked KMeans to find three clusters so that a comparison with the Mukherjee et al., (1998) results could be performed. (Mukherjee, Feigelson, Babu, Murtagh, Fraley, & Raftery, 1998).

The results indicate which input instance belongs to which class and a statistical breakdown of each class by feature value. The statistical breakdown of the classes formed by KMeans are presented in table 2. The column headings are GRB features.

Table 2. KMeans results

cluster	$\log T90$	$\log HR321$	$\log Fluence$
0	-0.186937	0.577752	-6.703456
1	1.214436	-0.007903	-6.123354
2	1.699839	0.308194	-5.264337

Since the Mukherjee et al., (1998) analysis did not use KMeans, the two outcomes differ. However, cluster 0 loosely corresponds to the *short* cluster, cluster 1 corresponds to the *intermediate* cluster, and cluster 2 corresponds to the *long* cluster.

5 FUTURE WORK

There are several issues that suggest further work on our ToolShed strategy. We describe initial thoughts on pursuing improvements.

5.1 Improve the scientist to software interaction

The process of implementing and refining our ToolShed prototype for GRB astronomers uncovered a significant issue that has yet to be addressed: *how should data be collected and/or selected to be placed into a ToolShed for scrutiny*? This question may seem obvious, but it is easy to underestimate the amount of work needed by a scientist prior to using automated data analysis tools. Also, it is easy to overestimate the quality of the results as they relate to the quality of the data gathered. The bottom line is that without sufficient work in the preparation of the data or without sufficient scientific understanding of the goals of the data analysis achieving meaningful results from following our proposed solution may be rendered impossible.

Our proposed solution lacks any automated strategy for helping the scientist with the preparation of the data. We have some minor features in our prototype that can help a scientist who is already familiar with our software. For example, it is easy to transform a feature (measurement) from its current form into a logarithm of the measurement. Computing ratios of two features is another example of easy data preprocessing. But much more research should be done to improve the automation of the data preparation aspect of scientific data analysis. Ideally, the software should recommend certain preprocessing computations along with rationale for the scientist to consider.

It has been estimated that as much as 80% of data analysis efforts can involve preprocessing, such as accounting for noise and dealing with missing information. Preprocessing may also include various data transformations such as data normalization and conversion as well as the addition and/or elimination of features. For analysis of scientific data, it is also necessary to understand the source of the data, such as the instrument taking the measurements and whether the instrument has intrinsic measurement biases. Both commercial and non-commercial tools are lacking in their ability to automate the identification and correction of errors in the data and to perform data transformation operations.

5.2 Automate the ToolShed setup process

One of the most arduous tasks imposed on the owner of the scientific data is to setup/configure the ToolShed around their data. Streamlining the process of installation and setup is essential to the success of our strategy.

5.3 2-stage mining

Evaluating unsupervised clustering is a more difficult task than evaluating supervised classification since the goals of an unsupervised clustering are often not as clear as the goals for supervised learning. In addition, the output of most unsupervised clustering algorithms provides little more than summary statistics about formed clusters.

Fortunately, supervised learning can help explain and evaluate the results of an unsupervised clustering. We call this process *2-stage mining* . Here is the procedure:

1. Assign each cluster a numeric value.
2. Create an output feature called “class.” For each instance, give the output feature the cluster number associated with the instance.
3. Build a supervised learner model with “class” designated as the output feature.

To illustrate, suppose we obtain three clusters using the K-Means algorithm. Following the 2-stage mining method, we assign each cluster an arbitrary number (e.g. 1, 2, 3) and label each instance according to the K-Means result. Next, we invoke a decision tree learner thereby creating a decision tree representing the formed clusters. As the decision tree gives us additional insight into the nature of the clusters, we are better able to interpret and understand what has been discovered. As an option, we can designate a subset of the clustering as training data and employ the remaining instances to test the supervised model for classification correctness.

The name "2-stage mining" comes from the fact that we are performing a second mining operation to understand better the output of an initial data mining session. We plan to automate this 2-stage mining process in the ToolShed.

5.4 Client/server model

Our initial architecture can be limiting when used as a central server with multiple simultaneous users running compute-intensive machine learning algorithms. We are working on a prototype of a system where the compute-intensive work is off-loaded to the client machine. Several approaches are possible. One approach involves implementing machine learning algorithms as java applets to run within the user's browser. This is easiest for the user but is lacking in efficiency and flexibility. A second approach is to require the user to install mining software on their machine. This software can be launched using the local operating system and would connect to the central server to download requested data. When the data is prohibitively large it may best to store the downloaded data on the client machine. With this approach, the central server becomes a software and database server.

There are many challenges associated with both of these approaches such as the ease of installation, ease of use by the client scientist, and portability of implementations. However, the potential benefits of a client/server architecture may be well worth the effort.

5.5 Mining Algorithms for Science Data

Looking at details of specific data mining algorithms is outside the scope of this paper. However, it is important to observe that the effectiveness of a ToolShed is limited by the mining algorithms it includes. There is a dearth of mining algorithms designed to work with measurement errors associated with some of the features. Unfortunately, scientific data typically includes measurably erroneous data. It may be reasonable to provide an existing data mining algorithm with an augmented data set where a measurement is supplied in one feature and the associated measurement error is supplied as another feature. Would it be more effective to modify an algorithm to learn differently based upon the measurement errors? For example, consider a clustering algorithm where some measure of instance similarity is used. If two instances have nearly identical feature values should higher measurement errors reduce their assessed similarity?

6 ACKNOWLEDGEMENTS

We are grateful to NASA for initially funding our project through the NASA Applied Information Science Research (AISR) program. Further funding was provided by the National Science Foundation grant. We also thank the anonymous reviewers for help us improve this paper.

7 REFERENCES

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, *39*(1), 1–38.

Fayyad, U., Haussler, D., & Stolorz, P. (1996a). Mining scientific data. *Communications of the ACM*, *39*(11), 51–57.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996b). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, *39*(11), 27–34.

Gennari, J., Langely, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, *40*, 11–61.

Grossman, R. L., Hornick, M. F., & Meyer, G. (2002). Data mining standards initiatives. *Communications of the ACM*, *45*(8), 59–61.

Haglin, D. J., Roiger, R. J., Hakkila, J., Pendleton, G., & Mallozzi, R. (2000). A GRB tool shed. In *Gamma-Ray Bursts: 5th Huntsville Symposium* (pp. 877–881) New York. AIP.

Hakkila, J., Giblin, T. W., Roiger, R. J., Haglin, D. J., Paciesas, W. S., & Meegan, C. A. (2003). How sample completeness affects gamma-ray burst classification. *Astrophysical Journal*, *582*, 320–329.

Hakkila, J., Haglin, D. J., Pendleton, G. N., Mallozzi, R. S., Meegan, C. A., & Roiger, R. (2000). Gamma-ray burst class properties. *Astrophysical Journal*, *538*, 165–180.

Hakkila, J., Haglin, D. J., Roiger, R. J., Giblin, T. W., Paciesas, W. S., & Meegan, C. A. (2001). An update on the GRB ToolSHED project status. In *Gamma-Ray Burst and Afterglow Astronomy* (pp. 556–558) Woods Hole, MA, USA.

Han, J., Altman, R. B., Kumar, V., Mannila, K., & Pregibon, D. (2002). Emerging scientific applications in data mining. *Communications of the ACM*, *45*(8), 54–58.

Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of Data Mining.* Cambridge, MA: MIT Press.

Jain, A., Mao, J., & Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *IEEE Computer*, *29*(3), 31–44.

Langley, P. & Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, *38*(11), 55–64.

Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, *28*(2), 129–137.

Mukherjee, S., Feigelson, E. D., Babu, G. J., Murtagh, F., Fraley, C., & Raftery, A. (1998). Three types of gamma ray bursts. *Astrophysical Journal*, *508*, 314–327. Preprint (astro-ph/9802085).

Paciesas, W. S., Meegan, C. A., Pendleton, G. N., Briggs, M. S., Kouveliotou, C., Koshut, T. M., Lestrade, J. P., McCollough, M. L., Brainerd, J. J., Hakkila, J., Henze, W., Preece, R. D., Connaughton, V., Kippen, R. M., Mallozzi, R. S., Fishman, G. J., Richardson, G. A., & Sahi, M. (1999). The fourth BATSE gamma-ray burst catalog (revised). *ApJS*, *122*, 465–495.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*, 82–106.

Quinlan, J. (1993). *C4.5: Programs for machine learning.* San Mateo, CA: Morgan Kaufmann.

Roiger, R. J. & Geatz, M. W. (2003). *Data Mining: A tutorial-based primer.* Boston, MA: Addison-Wesley.

Smyth, P., Pregibon, D., & Faloutsos, C. (2002). Data-driven evolution of data mining algorithms. *Communications of the ACM*, *45*(8), 33–37.

Weir, N., Fayyad, U. M., & Djorgovski, S. G. (1995). Initial galaxy counts from digitized POSS-II. *Astron. J.*, *109*(6), 2401–2412.

Weiss, S. M. & Indurkhya, N. (1998). *Predictive Data Mining: A practical guide.* San Francisco, CA: Morgan Kaufmann.