Developing innovative applications in agriculture using data mining

Sally Jo Cunningham and Geoffrey Holmes

Department of Computer Science
University of Waikato
Hamilton, New Zealand
email: {sallyjo, geoff}@cs.waikato.ac.nz

The WEKA (Waikato Environment for Knowledge Analysis) system provides a comprehensive suite of facilities for applying data mining techniques to large data sets. This paper discusses a process model for analyzing data, and describes the support that WEKA provides for this model. The domain model 'learned' by the data mining algorithm can then be readily incorporated into a software application. This WEKA-based analysis and application construction process is illustrated through a case study in the agricultural domain—mushroom grading.

keywords: machine learning, data mining, data analysis, application development

1. INTRODUCTION

Data mining is the process of discovering previously unknown and potentially interesting patterns in large datasets (Piatetsky-Shapiro and Frawley, 1991). The 'mined' information is typically represented as a model of the semantic structure of the dataset, where the model may be used on new data for prediction or classification. Alternatively, human domain experts may choose to manually examine the model, in search of portions that explain previously misunderstood or unknown characteristics of the domain under study.

In our work, we concentrate on machine learning techniques for inducing domain models or analyzing datasets (described further in Section 3). Machine learning algorithms provide models with a classification/prediction accuracy comparable to, for example, artificial neural networks, but which are more intelligible to humans than a neural model.

The WEKA[1] research team has two objectives: to mine information from existing agricultural datasets produced by New Zealand scientists and research organizations; and to perform basic research in data mining by developing new machine learning algorithms. To support these goals, we have developed a data mining workbench, the WEKA system, that incorporates the following tools: a set of *data pre-processing routines*, supporting the manipulation of raw data and its transformation into an appropriate form for data mining; *feature selection tools*, useful for identifying irrelevant attributes to exclude from the dataset; classifiers and other *data mining algorithms*, capable of handling categorical and numeric learning tasks; *metaclassifiers* for enhancing the performance of classification data mining algorithms (for example, boosting and bagging routines); *experimental support* for verifying the comparative robustness of multiple induction models (for example, routines measuring classification accuracy, entropy, root-squared mean error, cost-sensitive classification, etc.); and *benchmarking tools*, for comparing the relative performance of different learning algorithms over several datasets.

[1] Pronounced to rhyme with *Mecca*, the *weka* is a flightless bird with an inquisitive nature found only on the

Extensive, automatically generated documentation of the WEKA source is available to guide the user through interactions with the system. Earlier, Unix-based versions of the WEKA system are further described in Holmes et al, 1994; Garner et al, 1995. The current version of WEKA is implemented as a suite of Java class libraries. It is freely available on the World-Wide Web (http://www.cs.waikato.ac.nz/~ml/weka). The software accompanies a new text on data mining (Witten and Frank, 1999) which documents and fully explains all the data mining algorithms incorporated in WEKA. Application programs written using the WEKA class libraries can be run on any computer with a WWW browser.

The general process of data mining is described in Section 2. Section 3 describes the specific machine learning algorithms represented in the current version of WEKA, and Section 4 describes WEKA tools for supporting the data mining process model. A case study illustrating data mining using WEKA appears in Section 5, and Section 6 presents our conclusions.

2. DATA MINING PROCESS MODEL

In the course of this project we have analyzed over 50 real-world data sets, primarily agricultural data sets provided by research institutes and businesses in New Zealand. From this experience we have developed a process model for applying data mining techniques to data, with the goal of incorporating the induced domain information into a software module (Figure 1). The key points of this model are (Garner et al, 1995):

- *a two-way interaction between the provider of the data and the data mining expert*. Both work together to transform the raw data into the final data set(s) input to the machine learning algorithms — with the domain expert providing information about data semantics and 'legal' transformations that can be applied to the data, and the data mining expert guiding the process so as to improve the intelligibility and accuracy of the results.
- *an iterative approach.* Machine learning is an exploratory process; it generally takes several cycles through the process model to find a good "fit" between a representation of the data and a data mining algorithm. In addition, distinct attribute combinations run through different schemes can produce wildly different data models, even though the predictive accuracy of the results may be equivalent. These alternative views may provide valuable insights into patterns covering different subsets of the data.

In the model presented in Figure 1, activity flows in a clockwise direction. In the pre-processing stage, the raw data is firstly represented as a single table, as required by the data mining algorithms included in WEKA. This table is translated into the ARFF format, an attribute/value table representation that includes header information on the attributes' data types. The data may also require considerable 'cleansing', to remove outliers, handle missing values, detect erroneous values, and so forth.

At this point the data provider (domain expert) and the data mining expert collaborate to transform the cleansed data into a form that will produce a readable, accurate data model when processed by a data mining algorithm. These two analysts may, for example, hypothesize that one or more attributes are irrelevant, and set aside these extraneous columns. Attributes may be manipulated mathematically, for example to convert all columns containing temperature measurements to a common scale, to normalize values in a given column, or to combine two or more columns into a single derived attribute.

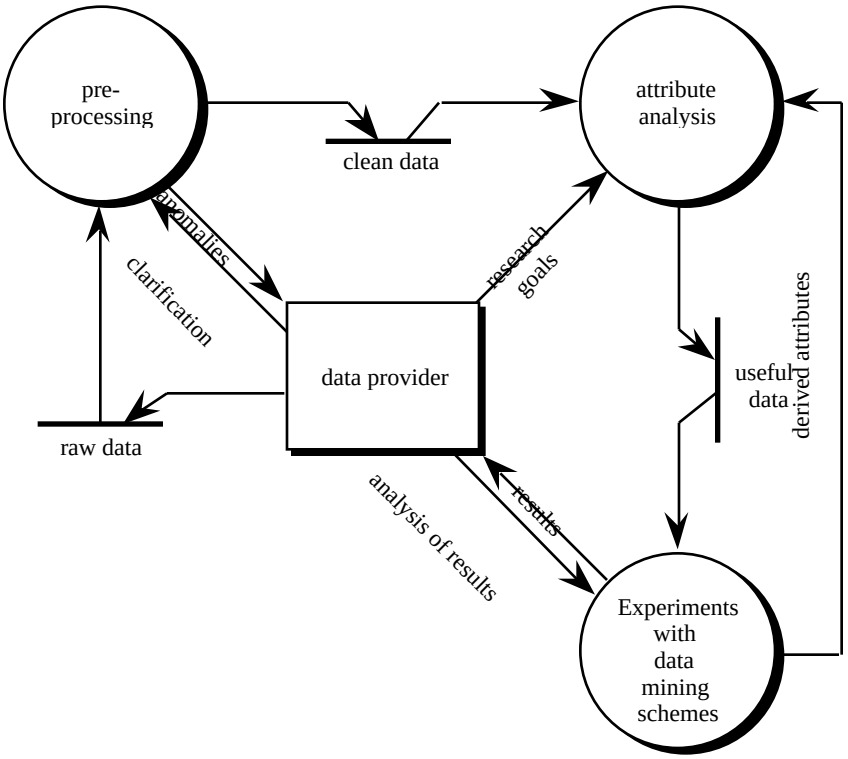

Figure 1. Process model for a machine learning application (data flow diagram)

One or more versions of the cleansed data are then processed by the data mining schemes. The domain expert determines which portions of the output are sufficiently novel or interesting to warrant further exploration, and which portions represent common knowledge for that field. The data mining expert interprets the algorithms' output and gives advice on further experiments that could be run with this data.

3. MACHINE LEARNING TECHNIQUES

The current version of WEKA contains implementations of twelve learning schemes: ten classifiers, a clustering algorithm, and an association rule learner. The software architecture is flexible enough to permit other learning schemes, and other types of learning schemes, to also be slotted into WEKA. In this section, we describe the types of learning that WEKA currently supports.

Classifiers

The output from this type of learning scheme is, literally, a classifier—usually in the form of a decision tree or set of rules that can be used to predict the classification of a new data instance. One attribute in the input table is designated as the category or class for prediction; the rest of the attributes may appear in the "if" portions of the rules (or the non-leaf nodes of the decision tree).

The most primitive learning scheme in WEKA is *ZeroR* (Table 1). This scheme models the dataset with a single rule. Given a new data item for classification, *ZeroR* predicts the most frequent category value in the training data for problems with a nominal class value, or predicts the average class value for numeric prediction problems. *ZeroR* is useful for generating a baseline performance that other learning schemes are compared to. In some datasets, it is possible for other learning schemes to induce models that perform worse on new data than *ZeroR*—an indicator of substantial overfitting.

weka.classifiers.ZeroR
weka.classifiers.OneR
weka.classifiers.NaiveBayes
weka.classifiers.DecisionTable
weka.classifiers.Ibk
weka.classifiers.j48.J48
weka.classifiers.j48.PART
weka.classifiers.SMO
weka.classifiers.LinearRegression
weka.classifiers.m5.M5Prime
weka.classifiers.LWR
weka.classifiers.DecisionStump

Table 1: The basic learning schemes in Weka

The next scheme, *OneR*, produces very simple rules based on a single attribute. *OneR* is also useful in generating a baseline for classification performance—indeed, this algorithm was found to perform as well as more sophisticated algorithms over many of the standard machine learning test datasets (Holte, 1993)! It appears that at least part of the reason for this result is that many of the standard test databases embody very simple underlying relationships in the data. Real world databases may also contain very simply structured information about a domain as well, and these simple relationships can be parsimoniously detected and represented by *OneR*.

NaiveBayes implements a Naïve Bayesian classifier, which produces probabilistic rules—that is, when presented with a new data item, the *NaiveBayes* model indicates the probability that this item belongs to each of the possible class categories (Langley et al, 1992). The Bayesian classifier is 'naïve' in the sense that attributes are treated as though they are completely independent, and as if each attribute contributes equally to the model. If extraneous attributes are included in the dataset, then those attributes will skew the model. Despite its simplicity, *NaiveBayes*, like *OneR*, can give surprisingly good results on many real world datasets.

DecisionTable summarizes the dataset with a 'decision table'. In its simplest state, a decision table contains the same number of attributes as the original dataset, and a new data item is assigned a category by finding the line in the decision table that matches the non-class values of the data item. This implementation employs the wrapper method (John and Kohavi, 1997) to find a good subset of attributes for inclusion in the table. By eliminating attributes that contribute little or nothing to a model of the dataset, the algorithm reduces the likelihood of over-fitting and creates a smaller, more condensed decision table.

Instance-based learning schemes create a model by simply storing the dataset. A new data item is classified by comparing it with these 'memorized' data items, using a distance metric. The new item is assigned the category of the closest original data item (its 'nearest neighbor'). Alternatively, the majority class of the *k* nearest data items may be selected, or for numeric attributes the distance-weighted average of the *k* closest items may be assigned. *IBk* is an implementation of the k-nearest-neighbours classifier (Aha, 1992). The number of nearest neighbors (*k*) can be set manually, or determined automatically using cross-validation.

j48 is an implementation of C4.5 release 8 (Quinlan 1993), a standard algorithm that is widely used for practical machine learning. This implementation produces decision tree models. *Part* is a more recent scheme for producing sets of rules called "decision lists", which are ordered sets of rules. A new data item is compared to each rule in the list in turn, and the item is assigned the category of the first matching rule (a default is applied if no rule successfully matches). This algorithm works by forming pruned partial decision trees (built using C4.5's heuristics), and immediately converting them into a corresponding rule.

SMO implements the "sequential minimal optimization" algorithm for support vector machines (SVMs), which are an important new paradigm in machine learning (Burgess, 1998). SVMs have seen significant application in learning models for text categorization (see, for example, Dumais et al, 1998). While SMO is one of the fastest techniques for learning SVMs, it is often slow to converge to a solution—particularly with noisy data.

WEKA contains three methods for numeric prediction. The simplest is *Linear Regression*. *LWR* is an implementation of a more sophisticated learning scheme for numeric prediction, using locally weighted regression (Atkeson et al, 1997). *M5Prime* is a rational reconstruction of Quinlan's M5 model tree inducer (Wang and Witten, 1997). While decision trees were designed for assigning nominal categories, this representation can be extended to numeric prediction by modifying the leaf nodes of the tree to contain a numeric value which is the average of all the dataset's values that the leaf applies to.

Finally, *DecisionStump* builds simple binary decision "stumps" (1-level decision trees) for both numeric and nominal classification problems. It copes with missing values by extending a third branch from the stump—in other words, by treating "missing" as a separate attribute value. *DecisionStump* is mainly used in conjunction with the *LogitBoost* boosting method, discussed in the next section.

Meta-Classifiers

Recent developments in computational learning theory have led to methods that enhance the performance or extend the capabilities of these basic learning schemes. We call these performance enhancers "meta-learning schemes" or "meta-classifiers" because they operate on the output of other learners. Instead of using a single classifier to make predictions, why not arrange a committee of classifiers to vote on the classification an instance? This is the basic idea behind combining multiple models to form an ensemble or meta classifier.

Two of the most prominent methods for constructing ensemble classifiers are boosting and bagging (Breiman, 1992). More often than not, these classifiers can increase predictive performance over a single classifier. However, the price for this increase in performance is that it is generally not possible to understand what is behind the improved decision making.

Both bagging and boosting vote on classifications using a weighted vote—each model in the ensemble predicts a class and assigns a confidence value to the prediction. These values are summed and the class with the largest value (most confidence) is chosen.

The two methods derive their collections of models of the data in quite different ways. Bagging works by building separate models of the training data using a sampling technique that deletes some instances and replicates others. In this way individual models are built separately with a fresh training set at each iteration (the number of iterations determines the number of models constructed for the ensemble).

Algorithmically:

Let n be the number of instances in the training data
For each of t iterations do
Randomly sample n instances (using deletion and replication)
Apply a learning technique to build a model from the sample
Store the model
End

Like bagging, boosting is iterative but instead of sampling fresh training data, each new model is influenced by the performance of those built previously. Instances that are incorrectly classified in previous iterations are promoted and those correctly classified are relegated. The key idea is to weight the instances and to use a learning algorithm that can take into account these weights when constructing its models. Initially, the weights are even and a model is constructed. The instances correctly classified by this model are given less weight so that the incorrectly classified instances will have more "importance" in the next iteration.

Algorithmically:

Assign equal weight to all instances
For each of t iterations do
Apply a learning technique to build a model from the weighted instances and store the resulting model
Down-weight each instance correctly classified by the model
End

The *AdaBoost.M1* (Freund and Schapire, 1996) boosting algorithm gives the user control over the boosting iterations performed. Another boosting procedure is implemented by *LogitBoost* (Friedman et al, 1998), which is suited to problems involving two-class situations—for example, the *SMO* class from above. In order to apply these schemes to multi-class datasets it is necessary to transform the multi-class problem into several two-class ones, and combine the results. The *MultiClassClassifier* boosting technique does exactly that.

Clustering

Clustering methods do not generate predictive rules for a particular class, but rather try to find the natural groupings (or "clusters") in the dataset. This technique is most often used in an exploratory fashion, to generate hypotheses about the relationships between data instances. Clustering is often followed by a second learning stage, in which a classifier is used to induce a rule set or decision tree that allocates each instance in the dataset to the cluster assigned to it by the clustering algorithm. These classifier-generated 'cluster descriptions' can then be examined to gain a semantic understanding of the clusters.

WEKA includes an implementation of the EM clustering algorithm. This algorithm makes the assumption, common to other clustering algorithms, that the attributes in the dataset represent independent random variables. Some clustering algorithms force each record to belong to exactly one cluster; EM permits an instance to belong to more than one cluster, a useful extension that, in practice, can support more flexible and more 'fuzzy' descriptions of the implicit structure of the dataset.

Association rules

WEKA contains an implementation of the Apriori algorithm (Agrawal, et al, 1993) for generating association rules, a type of learning scheme commonly used in "market basket analysis" (MBA). MBA algorithms have recently seen widespread use in analyzing consumer purchasing patterns—specifically, in detecting products that are frequently purchased together. These algorithms were developed in response to the vast flood of transaction data produced by barcode-based purchasing/ordering systems. This data was quickly recognized by the business world as having immense potential value in marketing, but traditional data analysis techniques could not cope with the size of the hypothesis space that these datasets engender.

For this type of analysis, data is logically organized into "baskets" (usually records in which the items purchased by a given consumer at a given time are grouped together). MBA algorithms such as Apriori discover "association rules" that identify patterns of purchases, such that the presence of one item in a basket will imply the presence of one or more additional items. A hypothetical example of such a rule might be that shoppers who purchase toothpaste are also likely to buy bananas on the same trip to the grocery store. This result can then be used to suggest combinations of products for special promotions or sales, devise a more effective store layout, and give insight into brand loyalty and co-branding.

4. WEKA TOOLS

In addition to the learning algorithms discussed above, WEKA also provides tools for pre-processing data and for comparing the performance of different learning algorithms.

Dataset pre-processing

WEKA's pre-processing capability is encapsulated in an extensive set of routines, called *filters*, that enable data to be processed at the instance and attribute value levels. These filters have a standard command-line interface with a set of common command-line options.

Many of the filter algorithms provide facilities for general manipulation of attributes—for example, to insert and delete attributes from the dataset. When experimenting with learning schemes in the development of a data mining application (Section 2), one of the most common activities involves building models with different subsets of the complete attribute set. WEKA provides three feature selection systems to aid in choosing attributes for inclusion in an experiment: a locally produced correlation based technique (Hall and Smith, 1998); the wrapper method (John and Kohavi, 1997); and Relief (Kira and Rendell, 1992).

In some cases it can be beneficial to apply a transformation function to an entire column in the dataset—for example, to normalize each value in an attribute. For nominal attributes, it may be advantageous to represent a multi-class attribute as a two-class attribute (thereby reducing the scale of the categorization problem to binary classification); alternatively, the number of classes may be reduced by merging two values of a nominal attribute into a single value. Filters are provided to support these transformations, which may be applied to non-class attributes as well. Since some learning schemes (eg, *SMO*) can only handle binary attributes, a filter is available that transforms a multi-valued nominal attribute into a binary valued attribute. Since many algorithms cannot handle real valued attributes, a

discretization filter is provided. It can perform unsupervised discretization (with equal width binning) or supervised discretization (using MDL; Fayyad and Irani, 1993).

Missing values occur frequently in real world datasets, and are difficult for many data mining algorithms to handle. Indeed, most algorithms simply omit lines of data containing a missing value—which can, in extreme cases, reduce the amount of useable data in a training set to the point that a reliable model cannot be formed. One technique for dealing with missing values is to globally replace them with estimated values before a learning scheme is applied; WEKA provides a filter that substitutes the mean (for numeric attributes) or the mode (for nominal attributes) for each missing value.

The presence of outliers in a dataset may seriously skew a model; a filter can remove outliers by deleting all instances that exhibit one of a particular set of nominal attribute values, or a numeric value below a given threshold.

Benchmarking algorithm performance

One of the key aspects of the WKA suite is the facility it provides to evaluate learning schemes consistently. For example, a researcher can create a "league table" summarizing the comparative performance of several schemes over a number of datasets. Table 2 illustrates the results of applying ten classifiers to 37 datasets from the UCI repository (Blake and Merz, 1998), a large repository of benchmark datasets for the machine learning research community.

W-L	Wins	Loss	Scheme
208	254	46	LogitBoost -I 100 Decision Stump
155	230	75	LogitBoost -I 10 Decision Stump
132	214	82	AdaBoostM1 Decision Trees
118	209	91	Naïve Bayes
62	183	121	Decision Trees
14	168	154	IBk Instance-based learner
-65	120	185	AdaBoostM1 Decision Stump
-140	90	230	OneR—Simple Rule learner
-166	77	243	Decision Stump
-195	9	204	ZeroR

Table 2: Ranking schemes

Column 2, *Wins*, is the number of datasets for which the scheme performed significantly better (at the 95% confidence level) than another scheme. *Loss* is the number of datasets for which a scheme performed significantly worse than another scheme. *W-L* is the difference between wins and losses to give an overall score. It would appear, for these 37 test sets, that Logit boosting simple stumps for 10 or 100 iterations is the best overall method among the schemes available in WEKA.

5. CASE STUDY: MUSHROOM GRADING

The data mining process model (Section 2) has been useful in focussing the analysis of real world datasets, using the WEKA analysis tools (Section 4) and the WEKA learning schemes (Section 3). In this section we describe one such application: a classification system for sorting mushrooms by grade (described in greater detail in Kusabs et al, 1998).

The goal of this project was to induce a classification system capable of sorting mushrooms into quality grades and achieving an accuracy similar to that attained by human inspectors. This research was carried out in collaboration with members of a local agricultural research organization (Postharvest Group, Ruakura Research Centre), who took the part of the data provider/domain expert in the process model (Section 2).

In this case, the data pre-processing stage involved not simply the cleansing of raw data, but the construction of a test dataset in collaboration with the agricultural researchers. This dataset contains descriptions of 282 mushrooms. The attributes included both objective measures (weight, firmness, percentage of cap opening) and subjective measures (Likert-scale estimates of the degree of dirt, stalk damage bruising, shrivel, bacterial blotch and *P. gingeri*). Three inspectors independently graded the mushrooms using the three broad commercial grades (1^{st}, 2^{nd}, and 3^{rd} grade).

In addition, digital images were captured for the 282 mushrooms. These images were analyzed to provide an additional 60 image-based attributes: frequency bin values (0-4) from the analysis of Red, Green and Blue (R,G,B) and Hue, Saturation and Value (H,S,V) histograms for top (t) and bottom (b) images of the sample mushrooms.

The wrapper method, in conjunction with model-building using the J4.8 classifier, proved useful in eliminating many of these 68 attributes (corresponding to the "attribute analysis" and "experiments with machine learning schemes" cycle in the process model, Figure 1). Using J4.8 and wrapper search, a separate model was developed for the three inspectors. The models developed suggested that each inspector used different combinations of attributes when assigning grades to mushrooms. All the predictive models used attributes from top and bottom images. Only Inspector 2 used weight for the classification of mushrooms into three grades. The subjective measurements (dirt, stalk damage, bruising, shrivel, bacterial blotch and *P. gingerii*) did not increase the accuracy of any of the prediction models, and so were eliminated by the wrapper technique. Finally, the models each incorporated between four and seven attributes—a significant reduction from the original 68! The average accuracy of the models was compared favorably with that of the human inspectors and the level of agreement with the human experts was, on average, acceptable.

These results indicate that visually-based attributes, which can be automatically extracted from digitized images, are sufficient for good separation of mushrooms into three broad quality bands (where 'good' is measured in comparison to human grading standards). The subjective attributes, commonly believed to play a crucial role in grading, are apparently irrelevant to the task. This surprising bit of 'mined' information echoes the conclusions of a classic machine learning paper (Michalski & Chilausky, 1980), which induced a set of rules for diagnosing soybean diseases that were strikingly dissimilar to expert opinions on the correct diagnosis procedure—but which were so accurate that one expert adopted the discovered rules in place of his own!

A more accurate, but humanly unintelligible, model was created using boosting. As discussed in Section 3, boosting is a "black box" approach for producing an ensemble of models that collectively achieve higher accuracy. In one experiment, 50 models were constructed and rated with AdaBoost. When making a prediction for new data, the individual models each have a vote proportional to their accuracy on the training data—hence the higher accuracy in grading mushrooms achieved by this model, but at the expense of its readability.

6. CONCLUSION

As illustrated by the case study presented in Section 5, information 'mined' from data can provide insights into the domain being studied that run counter to the received wisdom of a field. Locating these surprising or unusual portions of the model can be the focus for a data mining analysis, so that the results can be applied back in the domain from which the data was drawn. In this case, the results indicate that the subjective attributes for mushroom grading may not be useful in practice, and so perhaps they need not be measured or recorded. Criteria based on the attributes found in the J4.8 models may prove useful in developing more objective standards for quality classification and market pricing for mushrooms.

In other data mining applications, the goal might be to use a model predictively, to provide automated classification of new instances. In these applications, the learning component will likely be a small part of a much larger software system. Since WEKA learning schemes are accessible from other programs, a learning module can be slotted into a larger system with a minimum of additional programming.

Figure 2, for example, shows a WEKA applet based on a J4.8 mushroom grading model, as described in Section 5. Image processing a picture of a mushroom cap (at left in Figure 2) provides data for the model to differentiate between A, B and C grade mushrooms. Different models, as generated from WEKA, can be easily substituted into the applet as desired.

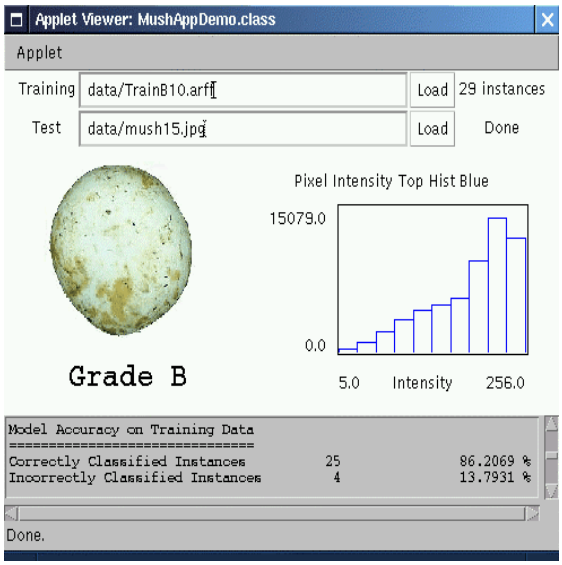

Figure 2: Mushroom grading applet

As the technology of machine learning continues to develop and mature, learning algorithms need to be brought to the desktops of people who work with data and understand the application domain from which it arises. It is necessary to get the algorithms out of the laboratory and into the work environment of those who can use them. WEKA is a significant step in the transfer of machine learning technology into the workplace.

REFERENCES

Agrawal, R., Imielinski, T., and Swami, A. (1993) "Mining association rules between sets of items in large databases." *Proceedings of the ACM SIGMOD Conference on Management of Data*, Washington, D.C., 207-216.

Agrawal, R., Imielinski, T. and Swami, A.N. (1993) "Database mining: a performance perspective." *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, 914–925.

Aha, D. (1992) "Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms." *International Journal of Man-Machine Studies*, Vol. 36, 267–287.

Atkeson, C.G., Schaal, S.A. and Moore, A.W. (1997) "Locally weighted learning." *AI Review*, Vol. 11, 11–71.

Blake, C.L. and Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Breiman, L. (1992) "Bagging predictors." *Machine Learning*, Vol. 24, 123–140.

Dumais, Susan T., Platt, John, Heckerman, David, and Sahami, Mehran (1998) "Inductive Learning Algorithms and Representations for Text Categorization." *Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM '98)*, Bethesda, MD, 148-155.

Fayyad, U.M. and Irani, K.B. (1993) "Multi-interval discretization of continuous-valued attributes for classification learning." *Proceedings of IJCAI*, Chambery, France, 1022–1027.

Friedman, J.H., Hastie, T. and Tibshirani, R. (1998) "Additive logistic regression: a statistical view of boosting." Technical Report, Department of Statistics, Stanford University.

Freund, Y. and Schapire, R.E. (1996) "Experiments with a new boosting algorithm." *Proceedings of COLT*, 209–217. ACM Press, New York.

Garner, S.R., Cunningham, S.J., Holmes, G., Nevill-Manning, C.G. and Witten, I.H. (1995) "Applying a Machine Learning Workbench: Experience with Agricultural Databases." *Proceedings of the Machine Learning in Practice Workshop*, 12th International Machine Learning Conference (Tahoe City, CA, USA). Available at <URL:http://lucy.cs.waikato.ac.nz/~ml/publications/1995/Garner95-imlc95.ps.gz>

Hall, M.A. and Smith, L.A. (1998) "Practical feature subset selection for machine learning." *Proceedings of the Australian Computer Science Conference*, Perth, Australia, 181–191.

Holmes, G., Donkin, A. and Witten, I.H. (1994) "WEKA: A Machine Learning Workbench." *Proceedings of the Second Australia and New Zealand conference on Intelligent Information Systems*, Brisbane, Australia. Available at <URL:http://lucy.cs.waikato.ac.nz/~ml/publications/1994/Holmes-ANZIIS-WEKA.ps.gz

Holte, R.C. (1993) "Very simple classification rules perform well on most commonly used datasets." *Machine Learning*, Vol. 11, 63–91.

John, G. H. & Kohavi, R. (1997) "Wrappers for feature subset selection." *Artificial Intelligence*, Vol. 97(1-2), 273-324

Kira, K. and Rendell, L.A. (1992) “A practical approach to feature selection.” *Proceedings of the 9th Int Conference on Machine Learning*, 249-256.

Kusabs N., Bollen F., Trigg L., Holmes G. and Inglis S. (1998) “Objective measurement of mushroom quality.” *Proceedings of the New Zealand Institute of Agricultural Science and the New Zealand Society for Horticultural Science Annual Convention*, Hawke’s Bay, New Zealand, 51.

Langley, Pat, Iba, W. and Thompson, K. (1992) “An Analysis of Bayesian Classifiers.” *Proceedings of the 10th National Conference on Artificial Intelligence*, 223-228, MIT Press.

Michalski, R.S. and Chilausky, R.L. (1980) “Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis.” *International Journal of Policy Analysis and Information System*, Vol. 4 (2), 125–161

Mitchell, Tom M. (1997) *Machine Learning*, McGraw Hill.

Piatetsky-Shapiro, G., and Frawley, W.J., eds. (1991) *Knowledge Discovery in Databases*. Menlo Park, CA, AAAI Press.

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

Wang, Y. and Witten, I.H. (1997) “Induction of model trees for predicting continuous classes.” *Proceedings of the Poster Papers of the European Conference on Machine Learning*, Prague, 128-137.

Witten, Ian H., and Frank, Eibe (1999) *Data Mining: Practical machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, San Francisco.

ACKNOWLEDGEMENTS

The Waikato Machine Learning group, supported by the New Zealand Foundation for Research, Science, and Technology, has provided a stimulating environment for this research.