Working Paper Series
ISSN 1170-487X

Machine Learning in Practice: Experience with Agricultural Databases

**by Stephen R. Garner,
Sally Jo Cunningham,
Geoffrey Holmes,
Craig Nevill-Manning &
Ian H. Witten**

Working Paper 95/13
May 1995

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Machine Learning in Practice: Experience with Agricultural Databases

Stephen R. Garner, Sally Jo Cunningham, Geoffrey Holmes,
Craig Nevill-Manning, and Ian H. Witten
Department of Computer Science
University of Waikato, New Zealand

1. Introduction

The Waikato Environment for Knowledge Analysis (WEKA[1]) is a New Zealand government-sponsored initiative to investigate the application of machine learning to economically important problems in the agricultural industries. The overall goals are to create a workbench for machine learning, determine the factors that contribute towards its successful application in the agricultural industries, and develop new methods of machine learning and ways of assessing their effectiveness.

The project began in 1993 and is currently working towards the fulfilment of three objectives: to design and implement the workbench, to provide case studies of applications of machine learning techniques to problems in agriculture, and to develop a methodology for evaluating generalisations in terms of their entropy. These three objectives are by no means independent. For example, the design of the WEKA workbench has been inspired by the demands placed on it by the case studies, and has also benefited from our work on evaluating the outcomes of applying a technique to data. Our experience throughout the development of the project is that the successful application of machine learning involves much more than merely executing a learning algorithm on some data.

In this paper we present the process model that underpins our work over the past two years for the development of applications in agriculture; the software we have developed around our workbench of machine learning schemes to support this model; and the outcomes and problems we have encountered in developing applications.

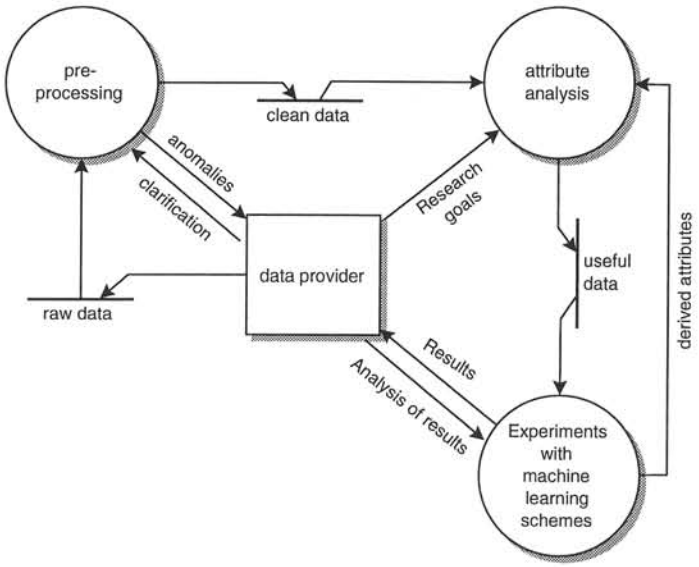

Figure 1. Process model for a machine learning application

2. Process Model

The process model we use for machine learning applications is presented as a data flow diagram in Figure 1. It is crucially dependent on two-way interaction between the provider of the data and the machine learning researcher, and enthusiasm on both sides plays a major role

[1] Pronounced to rhyme with "Mecca".

in ensuring that the model works. These two participants have quite different motivations for seeing the application through, and it is important that they are prepared to learn something from each other. The machine learning researcher, for example, must care more about finding something interesting in the data than building a new tool to analyse it. This requires discipline: tool-building is something that computer scientists gladly take on, whereas finding things in data involves liaison, listening, questioning, and perhaps failure—activities that they are not so comfortable with.

We receive the agricultural data that we analyse from either agricultural research institutes or private companies. In soliciting data, we find that we must first "sell" the concept of machine learning to research scientists, who tend to be wary of non-standard or non-statistical approaches. One useful technique is to stress the commonalities between statistics and machine learning, presenting them as complementary rather than promoting machine learning as a replacement for statistical analysis [Cunningham, 1995]. A second is to point out that machine learning is useful for generating hypotheses, which can then be supported by further statistical analysis or scientific testing.

In the initial visits to an agricultural institute, we emphasise the need to establish a partnership in analysing the data. We provide expertise in applying machine learning algorithms, but the data providers must supply domain information to effectively and efficiently guide analysis. Machine learning is not a "one-shot" operation; we generally must collaborate closely with those who gathered the data.

2.1 Preparation of the database—from raw to clean data

Having obtained the raw data, it must be massaged into a form suitable for processing by the automated tools. In the case of the WEKA system, the data is extracted and translated into a standard format we call ARFF, for Attribute Relation File Format (Holmes *et al,* 1994; McQueen *et al,* 1994). This generally involves taking the physical extract of a database and processing it through a series of steps to generate an ARFF data set. Anomalies arise in this process and must be resolved via consultation with the data provider. This may result in new data being generated, or in a better understanding of the values and attributes stored in the database.

The data must be transformed into the single flat table required by the algorithms in WEKA. Data is commonly obtained by retrieving it from a relational database using some form of query language (eg, SQL). This often involves taking data from a variety of sources or tables in the database and combining them into a single relation. This process, called denormalization, is the reverse of the process that was used to structure the database in the first place in order to impose integrity constraints, reduce update anomalies and eliminate data redundancy. The problems arising from denormalization are two-fold.

First, the data set that is extracted can be very large, in terms of both the number of columns or attributes and the number of rows or examples. The machine learning software and the computer system it is running on must both have sufficient resources to cope with the data. Second, denormalization may reintroduce functional dependencies between attributes that were the very reason for structuring the database in the first place. If these relationships are identified as patterns by the system, they will not be considered "interesting" by the user. Most machine learning tools will introduce these two problems because they only work with a single relation, though some techniques (eg, FOIL) use first-order logic and work with multiple relations.

Once the database is in a single relation, each attribute must be examined in order to determine its data type—for example, whether it contains numeric or symbolic information. Numeric values may include measurements, such as the diameter of an apple, while symbolic values could be the day of the week or whether a cow has had a calf this year. This information may be ascertained from the original database's data dictionary (if there is one), but there are a number of pitfalls.

Databases that have been designed for custom software may contain numeric values that are actually not to be treated as numbers but rather as codes describing a particular condition. For example, a field *production index* may have the value –1 to indicate that no measurement

was recorded. In such cases, one may replace the –1 with a "missing value" token, or if that is not possible then simply remove those records. Another example is a field in the database that is of type integer but whose contents are not used arithmetically. This may arise in the case of an *identification number* field, for which certain operations—such as taking the average of the field's values—are meaningless. Changing the field to one where the numbers are treated as nominal values will eliminate the possibility of the system creating inappropriate rules.

Conversely, symbolic values may have an associated ordering—for example, the days of the week—in which case it may be better to restructure the attribute into one that allows rules to be produced that express situations such as

working_day $\leq$ *Tuesday* rather than *working_day is Monday* or *working_day is Tuesday*.

Mapping the symbolic values to integers will allow such operations, though one must be careful not to permit arithmetic or statistical operations to be applied to the integers.

The user preparing the data set needs to be aware of co-dependent and implied attributes. The former occur when one attribute contains a measurement and another indicates how accurate that measurement is. The second attribute might be a necessary qualification on the first one—for example only use the measurement if its accuracy exceeds 95%. The latter occur when the absence of a value in one attribute implies its existence in another attribute. For example, in the cow culling data set described in Section 3, the absence of a value in the *transfer in date 3* field means check the *transfer in date 2* field to determine when the cow was transferred into the herd.

Missing values in the data set can create a variety of problems. Sometimes they merely indicate the absence of information; other times they actually convey information. In the first case all missing values should be replaced by a token recognised by the WEKA system. In ARFF, missing values are represented by "?". The presence of the missing value token means that the system will avoid creating rules for which the absence of a value determines the classification. On the other hand, a missing value may convey information—for example, in an attribute that contains disease information the absence of a value could signal *No disease*. Sometimes, both cases occur together in a single attribute, and it may not be possible to distinguish a missing value from a default one.

Real data is surprisingly dirty, and requires sustained effort to check for consistency in measurements, locate erroneous values, and determine sources of noise. Again, this process goes more smoothly when working closely with the data providers. As with any data analysis technique, the higher the data quality, the better the model will be. If many attribute values are missing or corrupt, the schemes may not have enough data to build a complete model. Two problems have been encountered while working with agricultural data sets.

The first is that classes may overlap when different classes have very similar attribute values. Figure 2 shows the results of evaluating a model describing apple bruise size created using C4.5 on a data set of 1440 instances. The model misclassified 404 apples in the data set, an error rate of 28.1%. The apples with tiny bruises seem to form a well-defined class, but the distinction between apples with medium bruises and those with larger bruises is not readily predictable using the attributes in the data set.

Classified as				*Actual*
tiny	small	medium	large	*Class*
302	11	2		tiny
10	41	59	1	small
	4	663	26	medium
		291	30	large

Figure 2. Classification accuracy matrix for apple bruising

A second problem occurs when one of several classes is much larger than the others. Known as the "small disjunct" problem, this may cause machine learning schemes to describe the data as a single class and treat the instances in the smaller classes as anomalies or errors. For

example, in one data set describing cows in heat, 98% of the examples are not in heat and the remaining 2% are not clearly distinguishable using the attributes in the data set. Thus the schemes come up with the general rule that a cow is not in heat, which is correct 98% of the time! Work is continuing in this area. One approach is to reduce the number of instances of the larger class in the training set while maintaining the numbers of instances in the smaller classes. Another is to construct a hybrid scheme by augmenting C4.5 with an instance-based learner [Ting, 1994].

2.2 Attribute analysis—from clean to useful data

At this point, we have cleaned up rows (data instances) and now need to determine the columns that are most likely to provide information and the one to be used for classification. The classification relates to the overall research goals of the data provider. These goals may have to be "extracted" from them from a knowledge of what types of thing can be done by the learning schemes. This extraction process is less painful if the data providers have been well briefed on the technology when the partnership was created.

Finding useful attributes is a problem of tractability: real datasets have thousands, perhaps millions, of records, each of which may have hundreds of fields. Some schemes cannot handle the huge numbers of hypotheses that must be calculated in analysing such a data set; others cannot handle them in a timely fashion. Pre-selection of potentially relevant attributes is critical, as omitting a key feature or including irrelevant ones can both lead to poor classification accuracy.

We run the data through one or more of the following schemes to select the most relevant features:

- 1R, the single-attribute classifier developed by Holte [Holte, 1993]. While Holte uses 1R as a stand-alone learning scheme, we view it as a feature selector. Applying 1R iteratively with each of the attributes in the raw data allows us to rank attributes by their classificatory power.
- C4.5, a similarity-based learning scheme that has proven in practice to be quite effective in winnowing out irrelevant attributes [Quinlan, 1992]. We apply C4.5 directly to the initial data table, and eliminate attributes that do not play a significant role in the resulting decision tree. As will be discussed in Section 4, we also gain quick insight into possible problems with the format of key attributes.
- FSS, the feature selection algorithm built into the MLC++ system [Kohavi et al, 1994]. FSS is a forward sequential selection algorithm which iteratively adds attributes and tests their effectiveness/relevance with an induction algorithm. It produces a list of what the induction algorithm considers to be the most relevant attributes.
- Runic, an algorithm for subset selection that examines rough numeric dependencies in the data [Smith and Holmes, 1995]. An attribute is considered potentially relevant if a particular subrange of its values can be used to predict the value of another feature more accurately than by pure chance. The degree of predictivity can be used to rank attributes, and the best predictors chosen are for analysis by machine learning algorithms.

2.3 Experimentation with machine learning schemes

The result of these processing steps is a table of (probably) useful attributes and consistent instances. This data is placed in an experimental loop across several schemes. In the absence of a comprehensive set of empirical tests to determine a single best learning algorithm to apply to a given data set, we find it most effective to apply more than one scheme. Different algorithms give different insights into the data set, and suggest further manipulations to perform on it.

We currently use the most common metric—classification accuracy—to determine the overall effectiveness of an experiment, and are developing entropy-based complexity measures for rule sets. In dealing with our data providers, we must also consider a metric that is not easily quantified: the "interestingness" of the derived rules. As we take our results

back to the domain experts, we are constantly reminded that it is less than impressive to proudly present them with common-sense facts from their field that we have calculated at great effort! A raw decision tree or set of rules needs to be processed by a human, to tease out the useful or novel information from the already-known.

One outcome of presenting results is the establishment of "derived attributes"—attributes that are used in practice but are not directly represented in the data. For example, *year of birth* may not be as useful as *current age*, if the data set includes several years of values; the production index of a cow relative to the herd average may well be more useful than the absolute production index, and so on. This step usually involves creating new columns from old ones. The most common operations include quantising a numeric attribute into intervals or classes; calculating a new attribute using a formula; and creating a new class attribute based on a series of logical conditions.

3. Software Tools

In this section we describe the tools we have developed to support our process model. These tools have evolved over time as we gained more experience with real datasets.

3.1 The WEKA workbench

The WEKA workbench has been designed to facilitate the processing of data from ARFF format to its final representation as a set of rules or decision tree. A variety of state-of-the-art, robust, and efficient machine learning schemes are provided for experimentation. The workbench is not, however, a multi-paradigm learner; that is, it does not combine machine learning techniques to produce new hybrid schemes. Instead, it concentrates on simplifying access to standard schemes so that their performance can be evaluated and compared.

The design of the workbench is something of a compromise between the needs of machine learning researchers and domain experts with end-user computing experience. We have tried to avoid alienating one or other of these groups of users. This usually means that we develop software that can be used either at the command line level, which is the preferred interface for expert researchers, or at the graphical user interface level, which is the preferred mode of access for domain experts and researchers new to the technology.

Papers describing the workbench have already appeared in the literature [Witten, 1993]. In this paper we concentrate on the supporting software that has been developed as a result of experience with real-world data.

3.2 Support for pre-processing and attribute analysis

Existing data must be converted to our standard format, ARFF, before it can be imported into WEKA. We have developed a collection of stand-alone programs that can be used to process standard formats such as INGRES files, but special-purpose code must be developed for obscure (or unique!) formats. Once the data is loaded into WEKA, it can be viewed with simple visualisation tools such as histograms and box plots. We also provide a spreadsheet for viewing data.

Data can be modified using the attribute editor, which provides facilities for creating derived attributes and generating new ARFF files (Figure 3). Formulae used to define derived attributes can be arbitrarily complex, and may apply logical and numeric operators to one or more existing attributes. In generating new ARFF files, the user can select subsets of both rows and columns from the original table. The attribute editor also allows the user to append comments recording the rationale behind new attributes.

Of the techniques for feature selection discussed in Section 2.2, only 1R and C4.5 are currently part of the workbench proper; we run FSS and Runic as stand-alone programs. We are currently considering how best to add these latter two algorithms to the workbench.

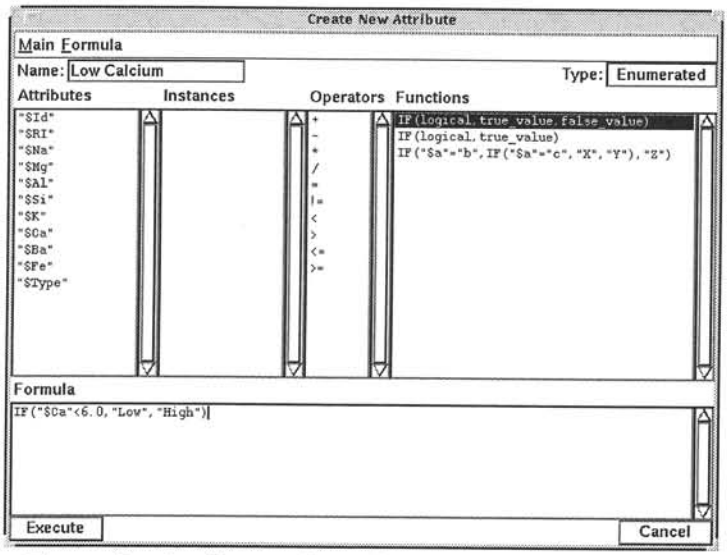

Figure 3. Attribute editor for new attribute creation

3.3 Support for experimentation

The experiment editor (Figure 4) is a simple graphical user interface tool for organising and running several machine learning schemes over several sets of data. It is a more general implementation of Holte's experimental paradigm [Holte, 1992].

A user specifies the ratio of the size of test and training sets and the number of runs required (Holte, for example, uses a 1/3:2/3 split of data and 25 runs). Data sets are chosen by the user, who is asked to specify the attribute to be used for classification. Schemes are chosen from those provided by the workbench. Once these items have been selected, the system randomly generates the required number of test and training files, and runs each of the selected schemes on this data. Results are collated in a text file and processed to provide summary statistics from the PREval evaluator (see below).

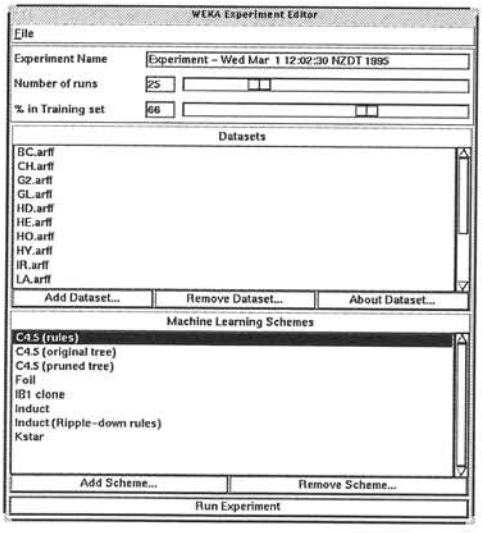

Figure 4. WEKA experiment editor

All output from a machine learning scheme is passed back to WEKA in the form of text, and can be displayed in a scrollable viewer. Text can be copied to other applications, printed, or saved to a file. A novel feature of the workbench is that it supplies a facility for "external" evaluation that can perform tests on the output from any machine learning scheme in a uniform way. If the user selects external evaluation, the output, as well as being displayed as text, is converted into an internal rule format and evaluated. The rule format is PROLOG-based, and rules can be executed using an evaluator called PREval. The precise details of

output translation varies for the individual schemes. For FOIL, which produces its output as rules anyway, it is a simple matter to convert the rule format to PREval. INDUCT, which produces rules in a special "ripple-down" structure, provides an option to produce its output directly in the PREval format. For C4.5, both decision trees and rules are converted automatically into PREval rules.

PREval takes a set of rules and an ARFF file, and evaluates how well the rules cover the classifications. It provides figures for classification accuracy, including the percentage correctly classified, incorrectly classified, classified by multiple rules, and not classified at all, as well as confusion matrices to show the distribution of misclassifications, and statistics on how well each rule does. PREval also incorporates an entropy measure for calculating both the complexity of rule sets, and the complexity of data sets with respect to a rule set.

4. An Agricultural Application of WEKA

In this section we examine a project in which WEKA has been used to find rules describing culling decisions in dairy herds. We relate our development of the application to the process model and the tools we described in sections two and three.

The "cow culling" database was provided by the Livestock Improvement Corporation in Hamilton, New Zealand. This institution operates a relational database system to track genetic history and production records of twelve million dairy cows and sires, of which three million are currently alive. Production data are recorded for each cow from four to twelve times per year, and additional information is recorded as events occur. Farmers receive information from the Livestock Improvement Corporation in the form of reports from which comparisons within the herd can be made. Two types of information that are produced are the *production* and *breeding indexes* (PI and BI respectively), which both indicate the merit of the animal. The former reflects the milk it produces with respect to measures such as fat, protein and volume, indicating its worth as a production animal. The latter reflects the likely merit of a cow's progeny, indicating its worth as a breeding animal. In a well-managed herd, the average value of these indexes will increase every year, as superior animals enter the herd and low-performance ones are removed.

One major decision that farmers must make each year is whether to retain a cow in the herd or remove it, usually to an abattoir. About 20% of the cows in a typical New Zealand dairy herd are culled each year, usually near the end of the milking season as feed reserves run short. The cows' breeding and production indexes influence this decision, particularly when compared with the other animals in the herd. Other factors that may influence the decision are:

- age: a cow is nearing the end of its productive life at 8–10 years;
- health problems;
- history of difficult calving;
- undesirable temperament traits (kicking, jumping fences);
- not being in calf for the following season.

The Livestock Improvement Corporation hoped that machine learning tools would provide insight into the rules that particular farmers actually use to make their culling decisions, enabling better information to be provided to farmers in the future. They provided data from ten herds over six years, representing 19 000 records each containing 705 attributes.

The initial data set we studied was extremely noisy, and indicated the need to include simple data visualisation techniques such as histograms in our workbench. A number of attributes contained a predominance of missing values; these could be quickly identified through visualisation, and were eliminated. These techniques are also helpful in determining viable clusters when discretizing real values. When the data was originally denormalized into a single table, the link and join relations duplicated some attributes (under different names). These were identified and deleted.

The principal tools used to select potentially relevant attributes were C4.5 (Quinlan, 1992) and FOIL (Quinlan, 1993). The initial flat table was run through C4.5 on the workbench.

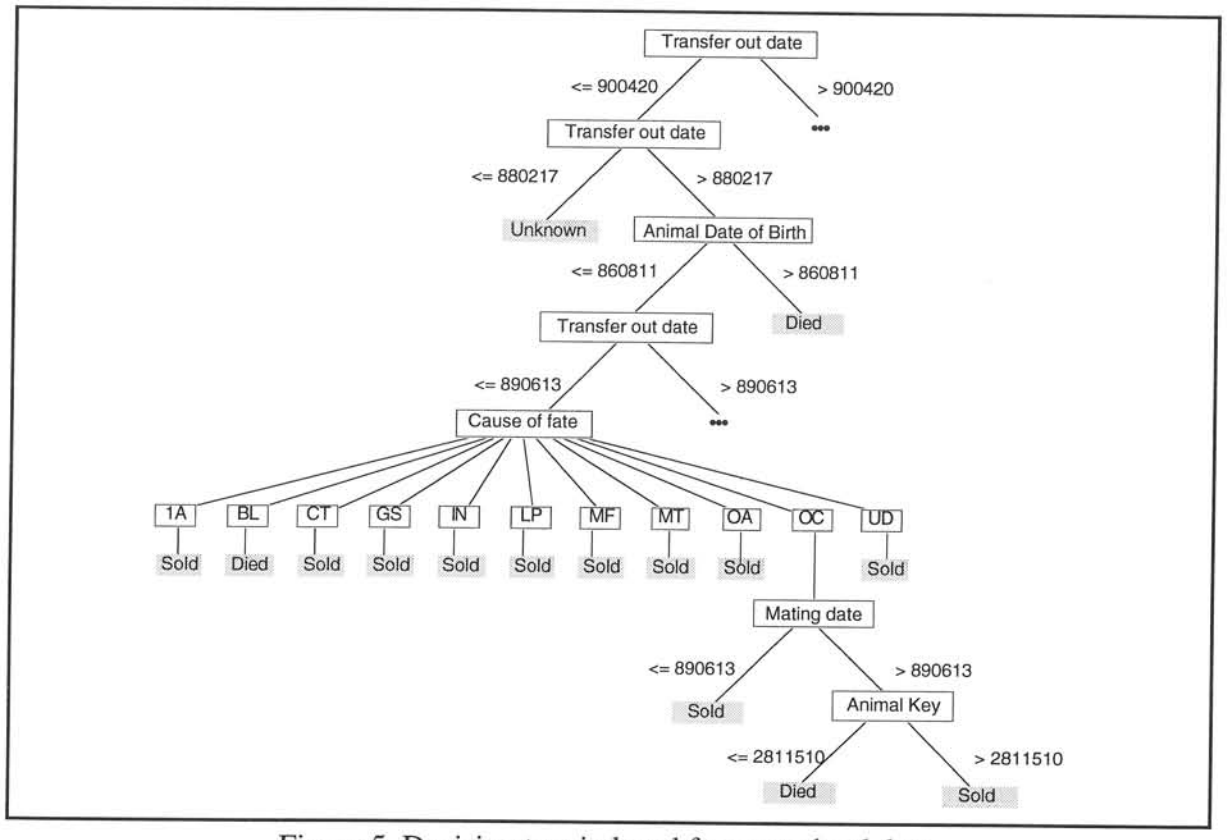

Figure 5. Decision tree induced from raw herd data

Cows were classified on their *fate code* attribute, which can take the values *sold*, *dead*, *lost* and *unknown*. The resulting tree, shown in Figure 5, proved disappointing.

At its root is the *transfer out date* attribute. This implies that the culling decision for a particular cow is based mainly on the date on which it is culled, rather than on any attributes of the cow. Next, the *date of birth* is used, but as the culling decisions take place in different years, an absolute date is not meaningful. A cow's age would be useful, but is not explicitly present in the data set. The *cause of fate* attribute is strongly associated with the *fate code*; it contains a coded explanation of the reason for culling. This attribute is assigned a value *after* the culling decision is made, so it is not available to the farmer who makes the culling decision. Furthermore, we would like to be able to predict this attribute—in particular the *low production* (LP) value—rather than include it in the tree as a decision indicator. Its presence artificially elevated the classification accuracy, predicting the culling decision correctly 95% of the time on test data. *Mating date* is another absolute date attribute, and animal key is simply a 7-digit identifier.

In discussions with staff from the Livestock Improvement Corporation, it was suggested that the culling decision may not be based on a cow's absolute performance, but on its performance relative to the rest of the herd. To test this, attributes representing the difference in production from the average over the cow's herd were added to the database. In order not to bias the learning process, the original attributes were retained, and the new attributes were not distinguished in any way—it was left to the machine learning schemes to decide if they were more helpful for classification than the original ones. This process involved close cooperation with Livestock Improvement Corporation. Discussions often ended up proposing more derived attributes, and clarifying the meaning of particular attributes. Staff were also able to evaluate the plausibility of rules, which was helpful in the early stages when the recreation of existing knowledge was a useful indication of the correctness of our approach.

It was necessary to create about forty new attributes, including a status code showing whether a cow is retained or culled and a variety of production and breeding indices relative to the

herd in a specific year. This experience was one of the primary motivating factors for the construction of the attribute editor described in Section 3.

After adding derived attributes, C4.5 produced the tree in Figure 6. The *fate code, cause of fate* and *transfer out date* have been combined into a *status code* which takes the values *culled* or *retained*. For each year, the records for cows that have previously been culled or have not yet been born are removed. Cows that were alive and were not transferred in that year are marked as *retained*, otherwise they are marked *culled*. If, however, a cow died of disease or some other factor outside the farmer's control, the record is removed. After all, the aim of this exercise is to discover the farmer's culling rules rather than the incidence of disease and injury.

The tree in Figure 6 is much more compact than the one in Figure 5. It was produced with 30% of the instances, and correctly classifies 95% of the remaining ones. The unconditional retention of cows two years or younger is due to the fact that they have not yet begun lactation, and so there is no indication of their productive potential. The next decision is based on the cow's worth as a breeding animal, which is calculated from the earnings of its offspring. The volume of milk that the cow produces is used for the final decision. This tree's decisions are plausible from a farming perspective, and its compactness indicates that it is a good explanation of the culling process. It is interesting to note that it consists entirely of derived attributes, further emphasising the importance of this step in the process.

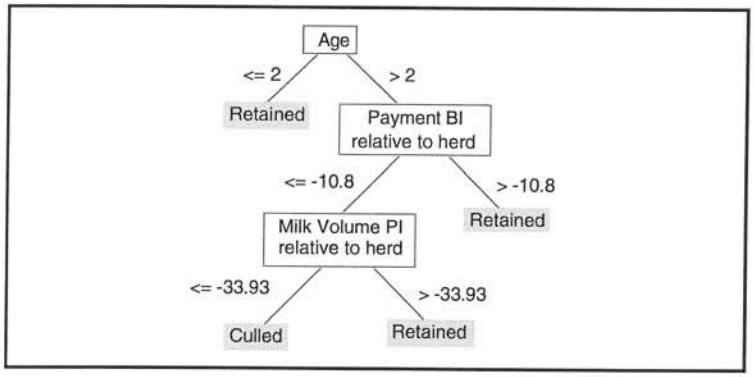

Figure 6. Cow culling decision tree

5. Summary

The discovery process is an iterative one. After taking a data set and analysing it, the results are used to augment the data set with new attributes or to adjust a scheme's parameters. Then the process is repeated. The WEKA workbench supports this process by providing the user with uniform facilities for manipulating data sets, configuring learning schemes and handling output. But while a good set of tools is important, forming a partnership with data providers is crucial. Domain knowledge is necessary to analyse data effectively, and our discussions with agricultural experts direct data processing, experimentation, and interpretation of results.

Our experience with New Zealand agricultural datasets has suggested several other improvements for the workbench. We tend to run a large number of experiments on a single dataset: creating new attributes, selecting subsets of rows and columns for processing, running the derived datasets across several machine learning schemes, testing different parameters on a single scheme, etc. Since it quickly becomes difficult to remember which combination of attributes produced what results on which algorithm, we are considering incorporating a scheme for organizing experimental results into the workbench. Similar facilities have recently been designed for statistics packages [Harner and Galfalvy, 1995; Young and Lubinsky, 1995], and it appears that these representation techniques are also applicable to machine learning analysis.

A longer-term goal would be to provide a method for codifying *a priori* domain knowledge and incorporating it into the machine learning model in a principled fashion. As discussed in Section 2, we use prior knowledge of the domain to create new (derived) attributes, to eliminate attributes that are unlikely to be useful, and to guide the experimentation process.

Most of this information is not recorded, or if recorded is not easily associated with the relevant dataset. We have implemented some simple solutions such as adding a comment field to the attribute editor so that we can keep track of what derived attributes represent. A much more difficult problem is to provide suggestions for creating derived attributes, or for effectively using domain information to incorporate a bias into the learning process.

References

Cunningham, S.J. (1995) "Machine learning and statistics: A matter of perspective.". *Working Paper Series 95/11,* Department of Computer Science, University of Waikato (Hamilton, New Zealand).

Harner, E.J., and Galfalvy, H.C. (1995) "Omega-Stat: an environment for implementing intelligenct modeling strategies." *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale (Florida, USA), pp. .252-258.

Holmes, G., Donkin, A., and Witten, I.H. (1994) "Weka: a machine learning workbench." *Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems,* Brisbane, Australia, pp. 357-361.

Holmes, G., and Nevill-Manning, C.G. (1995) "Feature selection via the discovery of simple classification rules." To appear in *Proceedings of Symposium on Intelligent Data Analysis (IDA–95)*, Baden-Baden, Germany, August, 1995.

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

Kohavi, R., John, G., Long, R., Manley, D. and Pfleger, K. (1994) "MLC++: A Machine Learning Library in C++," *Tech Report*, Computer Science Dept, Stanford University.

McQueen, R.J., Garner, S.R., Nevill-Manning, C.G., and Witten, I.H. (1994) "Applying machine learning to agricultural data." In Press, *Journal of Computing and Electronics in Agriculture.* Also available as *Working Paper Series 94/3* Department of Computer Science, University of Waikato (Hamilton, New Zealand).

McQueen, R.J., Neal, D.L., DeWar, R.E., and Nevill-Manning, C.G. (1994) "The WEKA machine learning workbench: its application to a real world agricultural database." *Proceedings of the Canadian Machine Learning Workshop*, Banff, Alberta, Canada.

Quinlan, J.R. (1992) *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Quinlan, J.R. and Cameron-Jones, R.M. (1993) "FOIL: a midterm report," *Proc European Conf on Machine Learning*, pp 3–20. Springer Verlag.

Smith, T.C., and Holmes, G. (1995) "Subset selection using rough numeric dependency." To appear in *Proceedings of Symposium on Intelligent Data Analysis (IDA–95)*, Baden-Baden, Germany, August, 1995..

Ting, K.M. (1994) "The problem of small disjuncts: its remedy in decision trees." In: Elio, R. (Editor), *Proc. Tenth Canadian Conference on Artificial Intelligence;* pp 91–97. Canadian Society for Computational Studies of Intelligence.

Witten, I.H., Cunningham, S.J., Holmes, G., McQueen, R., and Smith, L. (1993) "Practical Machine Learning and its Application to Problems in Agriculture." *Proceedings of the New Zealand Computer Society Conference*, Auckland, New Zealand, pp. 308-325.

Young, F.W., and Lubinsky, D.J. (1995) "Learning from data by guiding the analyst: on the representation, use, and creation of visual statistical strategies." *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale (Florida, USA), pp. 531-539.