The WEKA machine learning workbench:
Its application to a real world agricultural database

Robert J. McQueen, Donna L. Neal, Rhys DeWar, Stephen R. Garner and Craig G. Nevill-Manning

Department of Computer Science,
University of Waikato,
Hamilton, New Zealand.
Email: bmcqueen@waikato.ac.nz

1 Introduction

Numerous techniques have been proposed for learning rules and relationships from diverse data sets, in the hope that machines can help in the often tedious and error-prone process of knowledge acquisition. While these techniques are plausible and theoretically well-founded, they stand or fall on their ability to make sense of real-world data. This paper describes a project that aims to apply a range of learning strategies to problems in primary industry, in particular agriculture and horticulture.

New Zealand's economic base has historically been agricultural, and while this emphasis has decreased in recent decades, agriculture is still a vitally important part of the country's wealth. Dairy farming is in turn a large part of the agricultural sector, and the Livestock Improvement Corporation has a mandate to improve the genetics of New Zealand dairy cows. To this end, they collect and analyze a wide range of data on millions of cows and bulls.

Learning useful concepts from the gigabytes of data that the Livestock Improvement Corporation database contains involved two things: a diverse collection of analytical techniques with a consistent interface, and appropriate processing of the raw data, involving some domain knowledge.

This paper describes the machine learning workbench that has been developed to fulfill the first criterion, followed by the processes that were involved in the data pre-processing. The results are encouraging, and indicate that machine learning indeed has a valid role in large-scale agricultural problem solving.

The Machine Learning Workbench

The *Waikato Environment for Knowledge Analysis* (WEKA[1]) is a machine learning workbench currently being developed at the University of Waikato. Its purpose is to allow users to access a variety of machine learning techniques for the purposes of experimentation and comparison using real world data sets. The workbench currently runs on Sun workstations under X-windows, with machine learning tools written in a variety of programming languages (C, C++ and LISP). The workbench is not a single program, but rather a set of tools bound together by a common user interface.

WEKA currently includes seven different machine learning schemes, summarized in Table 1. In a typical session, a user might select a data set, run several different machine learning schemes on it, exclude and include different sets of attributes, and make comparisons between the resulting concepts. Output from each scheme can be viewed in an appropriate form, for example as text, a tree or a graph. To allow the user to concentrate on experimentation and interpretation of the results, they are protected from the implementation details of the machine learning algorithms and the input format that they require.

---

[1]   The Weka is a cheeky, inquisitive native New Zealand bird about the size of a chicken.

Scheme	Learning approach	Reference
Autoclass	Unsupervised Bayesian classification	Cheeseman *et al*. (1988)
OC1	Oblique decision tree construction for numeric data	Murthy *et al*. (1993)
Cobweb	Incremental conceptual clustering	Fisher *et al*. (1987)
C4.5	Supervised decision tree induction	Quinlan (1992)
CNF & DNF	Conjunctive and disjunctive normal form decision trees respectively	Mooney (1992)
Prism	DNF rule generator	Cendrowska (1987)
Induct	Improved Prism	Gaines (1991)
FOIL	First-order inductive learner	Quinlan (1990), Quinlan (1991), Quinlan *et al*. (1993), Cameron -Jones *et al*. (1993)

Table 1: Machine Learning schemes currently included in the WEKA workbench

The WEKA user interface was implemented using TK/TCL (Ousterhout 1994), providing portability and rapid prototyping. The main panel of the workbench is shown in Figure 1. On the left is the file name and other information about the current data set. The next column shows a list of all the attributes in the data set, along with information about the currently selected attribute. In the list, a filled box indicates an attribute that will be passed to the learning scheme; an empty box means that the attribute will be ignored. A filled diamond indicates that the learning schemes will attempt to classify on that attribute. In the third column, the values that this attribute can take are listed. If a particular value is selected, then rules will be formed to differentiate tuples with this value from those without. Otherwise, classification rules for each value will be generated. The fourth column lists the available machine learning schemes. Pressing a button marked '?' displays a short description of a particular scheme. In the rightmost column, the user can control the way that the data is viewed and manipulated.

To ensure input format independence, the data sets are converted to an intermediate format which includes information about the data set's name, attribute names, attribute data types, value ranges (enumerations for nominal data, intervals for numeric data), and the data itself. When a machine learning scheme is invoked, the data set is converted to the appropriate input form using a customized filter. A range of filters is also available for converting new files to the common format.

As well as the machine learning tools, the workbench is being expanded to support a variety of tools such as statistical analysis programs and spreadsheets, to allow the user to perform additional analysis, verification and data manipulation.

The Dairy Herd Data

The Livestock Improvement Corporation operates a large relational database system to track genetic history and production records of 12 million dairy cows and sires, of which 3 million are currently alive. Production data is recorded by LIC for each cow from four to twelve times per year, and additional data recorded as events occur. Farmers in turn receive information from LIC in the form of reports from which comparisons within the herd can be made. Two types of information that is produced are the *production* and *breeding indexes* (PI and BI respectively), which indicate the merit of the animal. The PI reflects the milk produced by the animal with respect to measures such as milk fat, protein and volume, indicating its merit as a production animal. The BI reflects the likely merit of a cow's
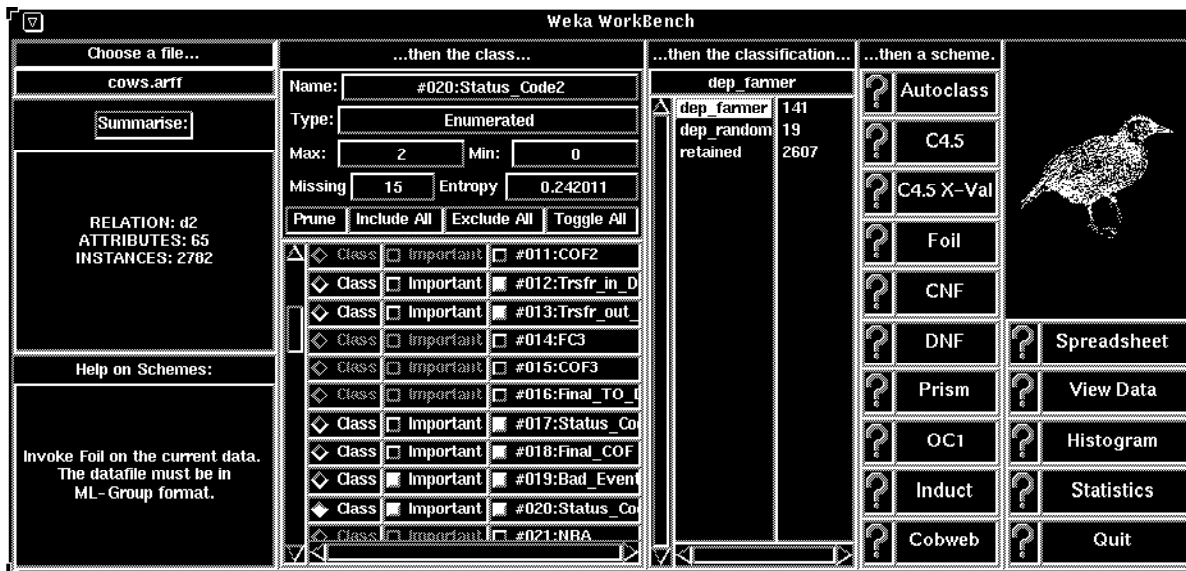
Figure 1: The WEKA user interface.

progeny, indicating its worth as a breeding animal.

One major decision that farmers must make each year is whether to retain a cow in the herd, or remove it from the herd, 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 cow's breeding and production indexes influence this decision, particularly when compared with the other animals in the herd. Other factors which may influence the decision could be:

- 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), and
- Not being in calf for the following season.

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

Problems with data sets for machine learning

When the initial unprocessed dataset as received from the Livestock Improvement Corporation was run through C4.5 on the workbench, the decision tree in Figure 2 was produced. Classification was done on the fate code attribute, which can take the values sold, dead, lost and unknown.

At the root of the tree 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 particularly meaningful. The cows *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
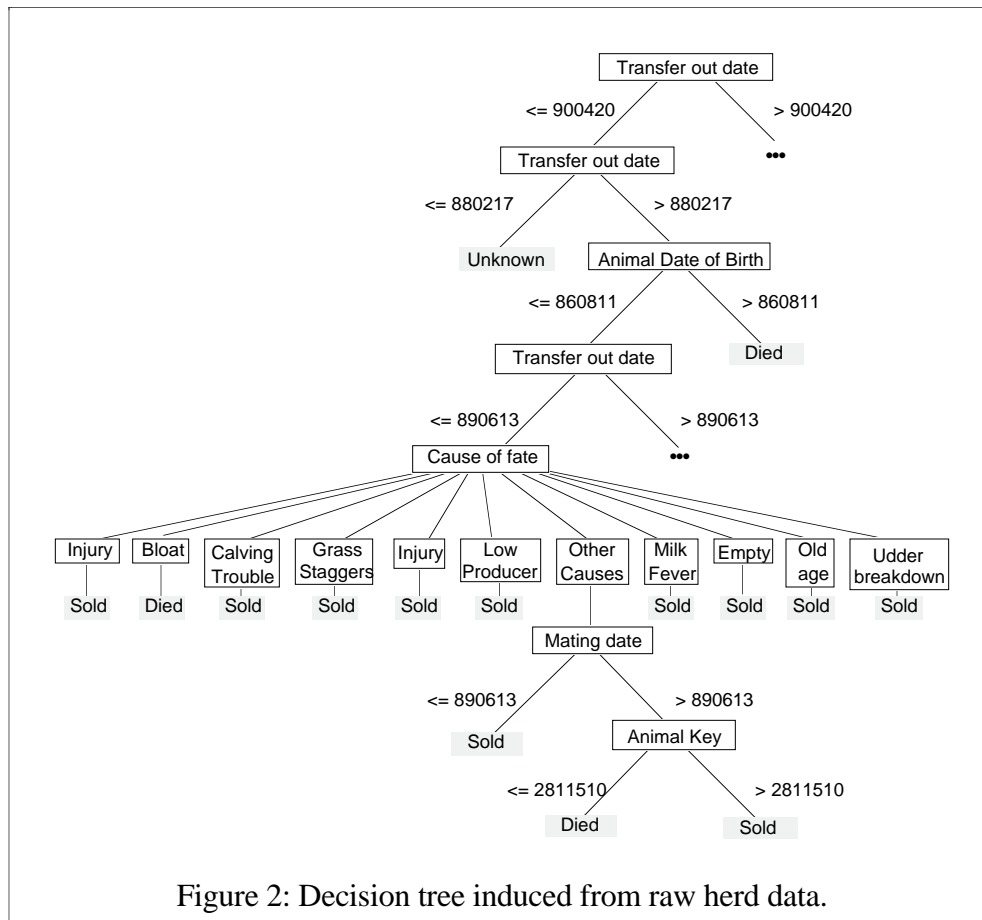
Figure 2: Decision tree induced from raw herd data.

attribute is assigned a value *after* the culling decision is made, so it is not available to the farmer when making the culling decision. Furthermore, we would like to be able to predict this attribute, in particular the *low production* value, rather than include it in the tree. This attribute made the classification accuracy artificially high, predicting the class 95% correctly on test data. *Mating date* is another absolute date attribute, and animal key is simply a 7-digit identifier.

The problems with this decision tree stem from the denormalization of the database used to produce the input, and the representation of particular attributes. The solutions to these problems are discussed below.

THE EFFECTS OF DENORMALIZATION

Many machine learning techniques expect as input a set of tuples, analogous to one relation in a database. Large databases invariably consist of more than one relation. The relational operator *join* takes several relations and produces a single relation from them, but this denormalizes the database, introducing duplication and dependencies between attributes. Dependencies in the data are quickly discovered by machine learning techniques, producing trivial rules that relate two attributes. It is therefore necessary to modify the data or the scheme to ignore these dependencies before interesting relationships can be discovered. In the project described here, trivial relationships (such as between the *fate code* and *cause of fate* attributes) are removed after inspecting decision trees by omitting one of the attributes from consideration.

In this particular data set, a more serious problem stemmed from the joining of data from several seasons. Each cow has particular attributes that remain constant throughout its

lifetime, for example *animal key* and *date of birth*. Other data, such as the number of weeks of lactation, is recorded on a seasonal basis. In addition to this, monthly tests generate production data, and movements from herd to herd are recorded at various times as they occur. This meant that data from several different years, months, and transfers were included in the original record which was nominally for one year; data that should ideally be considered separately (see Table 2).

While culling decisions can occur at any point in the lactation season, the basic decision to retain or remove an animal from the herd may be considered, for the purposes of this investigation, to be made on an annual basis. Annual records should contain only information about that year, and perhaps previous years, but not "foresight" information on subsequent data or events as may have been included through the original extract from the database. The dataset was renormalized into yearly records, taking care that "foresight" information was excluded. Where no movement information (which included culling information) was recorded for a particular year, a *retain* decision replaces the missing value. Monthly information was replaced by a yearly summary. While the data set was not fully normalized (dependencies between animal ID and date of birth still existed, for example), it was normalized appropriately for this particular application.

ATTRIBUTE REPRESENTATION

The absolute dates included in the original data are not particularly useful. Once the database was normalized into yearly records, these dates could be expressed relative to the year that the record represented. In general, the accuracy of these dates only needed to be to the nearest year, reducing the partitioning process evident in Figure 1.

In a discussion with staff from the Livestock Improvement Corporation, it was suggested that a 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 hypothesis, attributes were added to the database representing the difference of production data from production data averaged over the cow's herd.

In order to prevent overly biasing the learning process, all of the original attributes were

Relation	N⁰ of attributes	Recording basis
Animal Birth Identification	3	Once
Animal Sire	1	Once
Animal	6	Once
Test Number Identification	1	Monthly
Animal Location	3×6	When moved
Female Parturition	5	When calving
New Born Animal	3×4	When calving
Female Reproductive Status	3	Once
Female mating	10×3	When mated
Animal Lactation	60	Yearly
Test Day Production Detail	12×43	Monthly
Non production trait survey	30	Once
Animal Cross Breed	3×2	Once
Animal Lactation - Dam	12	Once
Female Parturition - Dam	5	Once
New Born Animal - Dam	3×4	When dam calves
Animal - Dam -Sire	2	Once

Table 2: Dairy herd database relations

Figure 3: Decision tree from processed data set
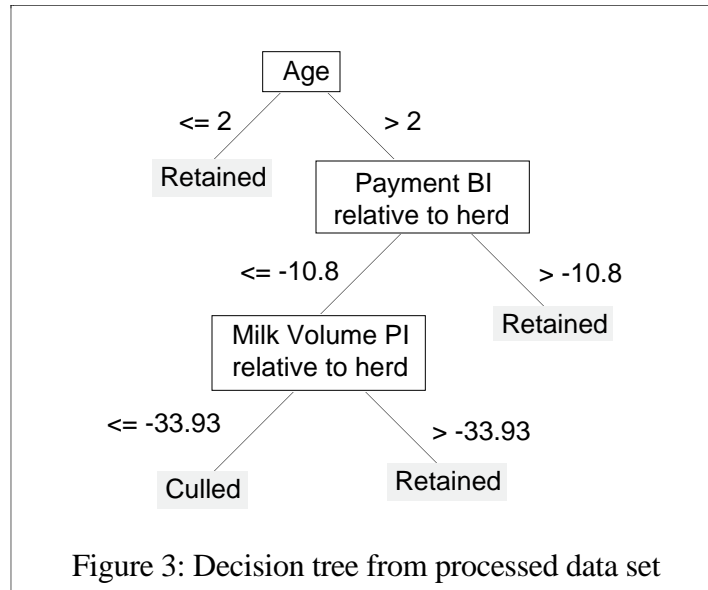
retained in the data set, and derived attributes were added to the records in an undistinguished way. It was left to the machine learning scheme to decide if they were more helpful for classification than the original attributes.

Throughout this process, meetings were held with staff at the Livestock Improvement Corporation. Discussions would often result in the proposal of more derived attributes, and the clarification of the meaning of particular attributes. Staff were also able to evaluate the plausibility of rules, which was helpful in the early stages when recreating existing knowledge was a useful measure of the correctness of our approach.

An obvious step would be to automate the production of derived attributes, to speed up pre-processing, and to avoid human bias. However, the space of candidate derived attributes is very large, given the number of operations that can be performed on pairs of attributes. Typing the attributes, and defining the operations which are meaningful on each type would reduce the space of possible derived attributes. For example, if the absolute dates in the original data are defined as dates, and subtraction defined to be the only useful operator on them, then the number of derived attributes would be considerably reduced, and useful attributes such as age would still be produced. This is an interesting and challenging problem that we may investigate in future.

5.3 SUBSEQUENT C4.5 RUNS WITH MODIFIED DATA

After normalizing the data and adding derived attributes, C4.5 produced the tree in Figure 3. Here, the *fate code, cause of fate* and *transfer out date* attributes have been transformed into an a *status code* which can take the values *culled* or *retained*. For a particular year, if a cow has already been culled in a past season, or if it has not yet been born, the record is removed. If the cow is alive in the given year, and is not transferred in that year, then it is marked as *retained*. If it is transferred in that year, then it is marked *culled*. If, however, it 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 diseases and injuries.

The tree in Figure 3 is much more compact than the full tree from which Figure 2 is taken. It was produced with 30% of the instances, and correctly classifies 95% of the instances.

The unconditional retention of cows two years or younger is due to the fact that they have not begun lactation, and no measurements of their productive potential have yet been made. The next decision is based on the cow's worth as a breeding animal, which is calculated

> culled(*tuple*) :- relative milk volume PI <= -34.2, relative payment PI <= -29.92
>
> culled(*tuple*) :- age > 9, relative fat BI <= -10.77, milk volume change <= -3.6, fat PI change > -4.2
>
> culled(*tuple*) :- relative fat PI <= -18.00, fat BI > 125, relative fat BI <= -5.39
>
> culled(*tuple*) :- protein PI change <= -7.6, protein PI <= 91.3, fat PI > 82.5
>
> *(relative = relative to herd, change = change since last season)*

Figure 4: Rules induced by FOIL on the processed data set.

from the earnings of the cow's offspring. The volume of milk that the cow produces is used for the final decision. The decisions in this tree appear plausible from a farming perspective, and the compactness and correctness of the tree indicate that it is a good explanation of the culling decision. It is interesting to note that the tree consists entirely of derived attributes, further emphasizing the importance of the pre-processing step.

5.4 RUNS WITH THE SAME DATA THROUGH OTHER WEKA SCHEMES

The data set contained many missing values, and many integer- or real-valued attributes. This reduced the number of schemes which were effective when used on this data set.

Next to C4.5, the next most useful scheme was FOIL (Quinlan 1992), which produces rules in the form of Horn clauses. The rules produced by FOIL were very similar to those produced by C4.5, including the removal of old or low-producing animals, and retention of high producers and young animals (Figure 4).

Both CNF and DNF (Mooney 1992) produced output similar to FOIL's. Induct (Gaines 1991) was limited by its inability to handle real and integer values but after hand-partitioning the ranges in the data it was possible to use it, though the results were less useful than those produced by C4.5 and FOIL.

6 Conclusions

Applying machine learning schemes to real-world data is difficult, mainly due to the representation of the data. The techniques described here of renormalizing data and deriving new attributes have succeeded in producing an interesting, plausible and compact decision tree from data which initially produced a much less useful tree. This is preliminary work, and it is hoped that more interesting results will follow on further exploration of the data, but the basic representation and procedural problems have been solved.

The ability to easily test various machine learning schemes on the same data has exposed flaws in several schemes, and further study on improving their performance will be investigated. The similarity of results from different schemes, however, has been encouraging.

Acknowledgments

References

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. Proceedings of the Fifth International Conference on Machine Learning (pp. 54-64). Ann Arbor, MI: Morgan Kaufmann.

Murthy, S.K., Kasif, S., Salzberg, S., & Beigel, R. (1993). OC1: Randomized Induction of Decision Trees. Proceedings of the Eleventh National Conference on Artificial Intelligence (pp. 322-327). Washington, D.C.

Fisher, D. (1987). Knowledge Acquisition Via Incremental Conceptual Clustering. Machine Learning, 2, (pp. 139-172).

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

Cendrowska, J. (1987) An algorithm for inducing modular rules. Int. J of Man-Machine Studies, Vol. 27, No 4, (pp. 349-370)

Gaines, B.R. (1991) The tradeoff between knowledge and data in knowledge acquisition in knowledge discovery in databases, (pp. 491-505), AAAI Press

Mooney, R.J. (1992) Encouraging Experimental Results on Learning CNF, *Technical Report, University of Texas*.

Ousterhout, J. K., Tcl and Tk toolkit, Addison-Wesley, 1994.

Quinlan, J.R. (1990), Learning Logical Definitions from Relations, Machine Learning 5, 239-266.

Quinlan, J.R. (1991), Determinate Literals in Inductive Logic Programming, Proceedings 12th International Joint Conference on Artificial Intelligence, 746-750, Morgan Kaufmann.

Quinlan, J.R. and Cameron-Jones, R.M. (1993), FOIL: a midterm report, 3-20, Proceedings European Conference on Machine Learning, Springer Verlag.

Cameron-Jones, R.M. and Quinlan, J.R. (1993), Avoiding Pitfalls When Learning Recursive Theories, Proceedings IJCAI 93, 1050-1055, Morgan Kaufmann.