WEKA MACHINE LEARNING PROJECT

COW CULLING

By: Rhys De War
Donna Liane Neal

September 1993 to March 1994

TABLE OF CONTENTS

TABLE OF APPENDICES

1. Introduction

This document describes the results of an investigation on using machine learning tools in an agriculural business context. This research was partially funded by a Foundation for Research, Science and Technology grant applied for in 1993/94.

The machine learning tools used were those presented under a common user interface by the WEKA unix workbench.

Data on dairy cows used supplied by the Livestock Improvement Corporation of New Zealand.

The chosen objective for testing machine learning and its application to agriculture, was to determine which data a farmer uses in the decision to cull a cow. Culling is the removing of a cow from the farm based on a subjective decision made by the farmer.

2. Objective

2.1 Project Objective

The objectives for the WEKA Machine Learning Project are outlined in the Foundation for Research, Science & Technology, Application for Funding, 1993/94 document.

The specific objective concerning the investigation was to provide case studies of the application of a similarity-based learning[1] techniques by firstly, identifying a problem suitable for tackling by similarity-based learning; and secondly, completing a case study of one of the problems selected and the extent to which the implementation techniques can be applied to it.

The problem relating to the agriculture industry suitable for tackling by similarity-based learning, is that of identifying the data which farmers use to determine which cows to cull. The term `cull' refers to a decision made by the farmer to remove a cow which always involves some subjective judgement.

2.2 Livestock Improvement Corporation Objective

The data providing the basis for the case project was obtained from the Livestock Improvement Corporation (LIC) of New Zealand. The Livestock Improvement Corporation (LIC) has a mandate to improve the genetics of New Zealand dairy cows. To this end, they collect, analyse and update farmers with a variety of production data on approximately 2 million cows, and maintain a large mainframe hierarchical database on these individual characteristics. The cows represented in the database are from 10,843 farms which accounts for 70 percent of New Zealand dairy farms. LIC were interested in supporting the investigation of machine learning techniques at the University of Waikato in the hope that

[1]Similarity-based learning can be defined as: given a set of objects which represent examples and counter examples of a concept, a similarity-based learner attempts to induce a generalized description that encompases all the positive examples and none of the counterexamples. T. H. Witten & B. A. MacDonald, p.79

the project might provide some insight on the data that was actually used by farmers in making their culling decisions, so that in future appropriate reports could be generated for the use of farmers in making better decisions.

3. Data

3.1 Data Source

As previously stated, the data was obtained from the Livestock Improvement Corporation of New Zealand database. The data was collected from 10 participant farms which contain cows reared from semen supplied by the Livestock Improvement Corporation. The farmer is contracted to supply data about the animals, so these datasets are less likely to contain missing values.

3.2 Data Structure

Two extracts were taken from the Livestock Improvement Corporation database in ASCII format. The first extract contained 705 attributes (refer Description of Extract File from Livestock Improvement, Appendix A)and came in the form of one text file for each participant for one year. (A total of 60 files since there were 10 participant farmers and and six years) The Test Day Production Detail (V262) data was contained in the second extract of 60 text files and needed to be merged in the position shown in Appendix A.

To understand the data more easily, the extracted data files can be thought of in a relational structure. This relational structure is shown in the diagram Relational Schema of LIC Database, Appendix B.

The number of records for each participant farmer for each year is shown in the following table:

Number of Records

Participant	1987	1988	1989	1990	1991	1992	Total
wxv	389	422	517	505	472	487	2792
bgpp	208	207	202	216	217	178	1228
bkyf	214	222	231	229	215	216	1327
fmtp	341	373	379	148	149	141	1531
dyvt	496	499	482	475	501	510	2963
cdjx	214	227	266	383	455	424	1969
btwq	280	316	341	329	357	379	2002
chgx	202	195	199	217	222	233	1268
wcr	323	328	323	349	344	316	1983
bkql	333	349	333	340	338	345	2038
Total	3000	3138	3273	3191	3270	3231	19103

NOTE: If the cow remained with the participant farmer for all six years, then the same cow would be featured in each of the six years records. (i.e. a record total of 2792 for wxv does not mean that wxv had 2792 individual cows)

4. Problems Encountered

4.1 General Problems of Machine Learning

Integer Attributes Which Contain Codes
Some problems arise when there are attributes in the database which have an integer value, but that value is actually a "code". For example, there may be an attribute in the database called "Primary_Language", which has integer values in the range 1 to 9. These values do not represent any quantity or measurement, but rather stand for the following languages;

1 =	English
2 =	Spanish
3 =	French
4 =	Portuguese
5 =	German
6 =	Russian
7 =	Chinese
8 =	Japanese
9 =	Afrikaans

A machine learning scheme has no knowledge of what these values stand for, it assumes that they are integers, with all the properties of integers. Classifications may be found based upon

Primary_Language > 4

which may be relevant, but if the data is sparse, or the frequency of some values is much greater than the frequencies of the others, then erroneous classifications may be arrived at. This is because, even though they are represented by an integer value, these languages have no inherent order, while integers do. Such attributes are better represented by Text fields, eg "English", "Spanish", etc, where each value is considered independent of the others, and no ordering is assumed.

This can commonly occur when an integer key is used as a unique identifier (eg, part_number), where the number is only intended to uniquely identify the part, and not as a quantity. However, sometimes, there is at least a partial ordering (eg, newly introduced parts have a higher part number).

Attributes Which Are Co-Dependent
Sometimes attributes are present whose meaning is reduced or entirely lost without the presence of another attribute. For example, we may have an attribute called "Engine Performance" and another called "Performance Certainty". The first is an arbitrary (numeric) value, indicating how well an engine performs. However, the second is a percentage certainty of how accurate the Engine Performance value is. These values may be present;

Mean Engine Performance	Performance Certainty
123.8	65%
119.0	80%
150.5	5%
95.1	100%

The first two instances may represent engines which have been built and have been tested, but are still undergoing refinement and improvement. Their performance values may change over time, and their Performance Certainty values will steadily increase. The fourth is an engine which has, perhaps, been a standard race proven engine for which the performance specifications have been established - hence the 100% certainty in the Mean Performance value - but which has now become outdated. However, the third is possibly an engine with a radical new design which is still in the prototype stage, and the performance estimates are still entirely theoretical. It's performance values may change quite drastically (either up or down) as testing proves or disproves it's theoretical capabilities, and changes are made to the design.

The problem we face is that the Performance Certainty is meaningless on it's own; it is co-dependent with the Mean Engine Performance. The Mean Engine Performance can be used for classification, but some of it's meaning is dependent on the Performance Certainty field. We may have two instances with the same performance values, but with vastly different certainties.

Implied Attributes
Most machine learning schemes assume that the attributes are independent of each other. However, in most databases, this is not the case. Some combinations of attributes may in fact form distributed representation of an implied attribute. For example, in our dataset we had attributes called "Transfer in Date" 1, 2 and 3, "Transfer out Date " 1, 2 and 3 and "Cause of Fate" 1, 2 and 3.

TID1	TOD1	COF1	TID2	TOD2	COF2	TID3	TOD3	COF3
870606	900105	OtherCauses	900105	?	?	?	?	?
860427	910207	OtherCauses	900207	900410	LowProducer	?	?	?
870319	910305	OtherCauses	910305	910416	OtherCauses	910416	?	?
870319	880205	DiedInjury	?	?	?	?	?	?

These combinations of Transfer in and out dates, along with the cause of fate determine whether the cow is present in a given year, and the reason the cow eventually left the herd if it is not still present.

To determine if the cow is present in a give year, we first look at its TID1 (Transfer In Date 1). If this is higher than the year we are concerned with, then the cow is not present as it has not yet joined the herd. Otherwise, we then look at transfer in date 3. If this value is not missing, we look at Transfer out date 3. If this is lower than the year we are concerned with, then the cow is not present, as it has already left the herd. If it falls within our year of concern, then it has left the herd within this year, and we take COF3 as it's reason for leaving. Otherwise, if it is greater than the year we are concerned with, then the cow remained in the herd this year, and we list it as being retained. However, if TID3 is missing, we look at TID2, and repeat the above for TID2, TOD2 and COF2. If TID2 is missing, we then look at TOD1 and COF1.

For example, in our example database above, assume we are interested in which cows were present in the herd in 1990. The first cow listed arrived in April 1987 (TID1) so we look for TID3. There is no TID3 so we look for TID2. TID2 is present, so we look at TOD2. This is missing, so we know the cow has not permanently left the herd. We list it as retained. The second instance arrived in 1986, but again, TID3 is missing. TID2 is present, and looking at TOD2, we see that the cow leaves the herd permanently in 1990. We list it as being removed from the herd this year, and list it's cause of fate as "LowProducer" (COF2). The third cow arrived in 1987, and TID3 is present. There is no TOD3, so the cow is present, and we list it as retained. The fourth cow arrived in 1987, but there is no TID3 or TID2. Looking at TOD1, we see that it left the herd in 1988 due to an injury, so we omit this cow from our 1990 data.

This walkthrough of a database query will seem fairly straightforward for anyone familiar with databases of any kind. However, a machine learning scheme with no knowledge of how to derive implied attributes from implicit attributes has no way of determining if a cow is present in a particular year or not. The scheme may be using data on cows which have been dead for years, or which have not even been born yet!

It can be argued that such implied attributes should be discovered by the schemes as patterns in the values of the attributes which imply them. However, if the data is sparse (ie, does not contain sufficient numbers of differing instances to derive the implied attribute), or the derivation of the implied attribute is complex, they may not be found by the schemes. However, if we already know that the implied attribute exists, there is no point making the scheme find it. We want to use the schemes to discover classifications that we do not know exist, or merely suspect (although confirmation of what we do know tends to strengthen faith in the schemes!).

Misrepresentation of Missing Values
Some attributes have a preset default value, which, if no other value is entered, is assumed to be the value of the attribute. However, if this is not explicitly stated, problems can arise, due to the way in which most machine learning schemes handle missing values. For example, if we have a field called "Disease". This can have the following possible values;

Tuberculosis

AIDS
Influenza
Glandular Fever
Meningitis

However, this set of values does not allow for a healthy case. This is implied by no entry being made for disease. This is perfectly understandable for a human user, but a machine learning scheme interprets the missing values differently. The scheme assumes that the missing value should have one of the other values it has seen. So in this case, it assumes that a disease is present in every instance, although it does not know what the disease is. What is required is a sixth explicit value "NoDisease" to prevent this form occurring.

The opposite problem occurs when there are values where there should be none. For example, we may have in our database a field called "Size". This has the following possible values;

Small
Medium
Large
Not_Recorded

Even though "Not_Recorded" is listed as a legal value for this attribute, we do not want classifications made on the basis of the size of a case not having been recorded. The "Not_Recorded" values need to be changed to missing values.

Numeric Values Which Are Misleading
This problem occurs in numeric fields when a particular value actually stands for a code, rather than an actual value. For example, if we have a field called "No. of Diseased Cells", we may find
a set of instances such as;

No. of Diseased Cells
1000
904
0
-1
0
15

Clearly, there can never be a negative number of cells present, so what does the -1 mean? This may be a code indicating that there was no test performed, or that the test results were biased, incorrectly acquired, etc. A human user may be able to interpret this, but an ML scheme treats this as a legitimate value for the field. If the fact that a test has been performed incorrectly is not important to the classification, this can be solved by replacing all the -1 values with missing values.

When performing statistical analysis on the data, these values will also tend to drag down means and increase standard deviations, etc.

Enumerated Types Which Are Ordered
Machine Learning schemes have no knowledge of order present in Text fields, unless supplied in advance with it. Most schemes will treat the order of the values as the order in which they are encountered as it scans the instances. For example, if we have a "Day of the Week" field with possible values;

Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday

a scheme has no concept of Tuesday being "before" Wednesday or "after" Monday. It will treat all these values as distinct and unrelated, and will not produce classifications based on

"Day of the Week" > Friday

for example. This may not pose a problem, if there are enough instances to learn that Saturday and Sunday both have a high information gain, where classifications may be made based on

"Day of the Week" = Saturday and
"Day of the Week" = Sunday

However, if it does pose a problem, for example if there are not enough instances for this trend to be shown, this can be solved by using integer values instead of text values.

4.2 Preprocessing the Data

In order for the schemes to produce meaningful results from the data, we had to ensure that the data we were presenting to them had the correct meaning. Some of the problems we had to overcome were;

Integer Attributes Which Mean Codes
The schemes interpret integers as integers, and have no way of determining that individual numbers stand for codes. These attributes were changed to enumerated text attributes. e.g. Reproduction Status.

Co-dependent Attributes
Some attributes in the dataset had no meaning in themselves, but instead were a measure of accuracy of another attribute. In this dataset, we were able to ignore these attributes, as they were irrelevant to the classification. e.g. Fat PI Reliability

Interpretation of Missing Values
In some attributes in the dataset, a missing value actually implied a default value, and we had to make sure that this was altered so that the default value was explicitly included. e.g. a Cause of Fate missing value implied a retained cow.

Absence of Missing Values
Other attributes had a legal value which actually indicated a missing value, and in some numeric attributes, this caused mean calculations to be incorrect. These were replaced with missing values.

4.3 The Classification Attribute

Initially, we used the Fate Code as our classification attribute. This seemed to be exactly what we were looking for. The Fate Code attribute contains a code indicating the "fate" of the animal, including a code, "C", indicating that the animal had been culled.

Seeing this, we were somewhat confused that only 2 of 389 cows in the first set of data we converted had fate codes of "C". However, we learned that other codes, such as sold, also indicated the farmer's decision to remove the animal form the herd.

The first problem encountered using this attribute was that there was no code indicating a "null fate", ie, that nothing had happened to the animal to cause it to leave the herd. Instead, an animal which was still in the herd had a blank value in this attribute. This meant that some of the workbench schemes, notably C4.5, would interpret a blank value for Fate Code as missing, and assume that it should have one of the other values possible for that attribute. Essentially, this meant that all cases were assumed to have left the herd. Initially we got around this by allowing "?", the ARFF representation of a missing value, as a legitimate value for fate code.

The next problem was that there were, not one, but three Fate Code attributes in the extract. The first one had the most values entered, and it was this Fate Code we used as our classification. The other two consisted mainly of missing values. These three attributes were to record movement of the animal, and sometimes these movements occurred within the herd. This meant that an animal may have a value for any of the Fate Code columns, but still be in the herd. We quickly decided that Fate Code was not a suitable classification attribute. We also tried the attribute Cause Of Fate, but quickly discarded this too, as it was closely related to Fate Code.

Initial Runs Using Fate Code

While Fate Code was not a good classification attribute given what we wanted the workbench to learn, our runs through the schemes using it as the classification gave us some insight into the structure of the data. We found that many splits were made on the attribute Cause Of Fate. This seemed to imply a strong link between the two. In fact, there IS a strong link: Fate Code records what happened to the animal, and Cause of Fate records why it happened. Generally, (missing values aside), an animal with a value for Fate Code has a value for Cause of Fate. An animal with no Fate Code should have no Cause Of Fate. Thus, the workbench was producing classifications based upon features of the LIA database, rather than upon features of the cows.

Also, the Cause Of Fate attribute provided an "easy way out": Cause of Fate is a derived attribute, based upon the farmer's reasons for removing that animal from the herd. The workbench schemes would produce rules such as;

```
Cause_Of_Fate = LP
        class S
```

Here, LP stands for low producer, and S stands for sold. This rule says that farmers sell low producing cows. This is a valid and sensible rule - however, it does not give any indication as to how to decide if a cow is a low producer. In practice, the farmer has decided that the animal is a low producer by observing the volume of milk produced by the cow. We wanted the schemes to learn how to make those decisions.

We also found that a high proportion of the values recorded for Fate Code were "G", meaning "Gone - unaccounted for", and that a similar proportion of the Cause Of Fate values were "Other Causes". We suspected that these values were used as catch-all categories, when a farmer could not remember what had happened to an individual animal, or why. This reduced the value of these two attributes, as they contained what could be called "artificial classes" which were useless for what we wanted the workbench to achieve.

Next, we found some discrepancies in the data. We examined the data, which was grouped by year. In the 1989 data, for example, we would find cows with a Fate Code of "S", but whose Transfer Out Date (the date of the animal's removal from the herd) was in 1992. We asked LIA why this might be so, and found that we had misinterpreted the structure of the extract data.

Where we had assumed that the Fate Code was functionally dependent on the Animal Key and the Year of the extract data, we found that the Fate Code was functionally dependent only on the Animal Key. So, if a given animal occurs in more than one year, it will have the same Fate Code in both records, regardless of whether it was removed from the herd in any of those years or not.

We also learned that the Transfer In Date and Transfer Out Date attributes recorded the dates of an animal's arrival in or removal from the herd. We had previously been suspicious of these dates, as there were again three attributes called Transfer In Date, and three called Transfer Out Date, which were associated with the Fate Code and Cause Of Fate. Also, most of the schemes required an enumerated attribute as a classification rather than a number. We used the Transfer In and Transfer Out Dates to derive a new attribute called Status Code, which indicated whether a cow stayed with or was removed form the herd in a given year.

Once the status code had been created, we essentially had a binary classification - either the case was Retained or Departed. However, there are many reasons for which a cow may leave the herd. A cow may be sold because it is a low producer, or it may have died due to an injury. Both of these cases would be recorded as Departed, but they are clearly different reasons for departure from the herd. Runs made using status code would correctly classify all the retained cows, but misclassify some departed cows as retained. Also, the rules/splits for classifying cases were made on strange attributes. We suspected this was because there was no differentiation between cows which had died naturally, and those which had been deliberately removed.

For this reason, we wanted to separate Status Code further, into Retained, Departed_Random and Departed_Farmer. Departed_Random would indicate that the cow had left the herd due to some random event such as injury or illness. Departed_Farmer would indicate that the cow had left the farm due to the farmer's decision to sell or cull it. Early runs using these codes correctly classified all the Retained cows, and all the Departed_Random. However, most of the Departed_Farmer were still classified as Retained.

4.4 Relational Database to Flat Form Dataset

With the complex structure of a relational database comes problems associated with extracting a two-dimensional dataset. Some of the problems are as follows:

Missing Values
When database relations are joined to form a flat file dataset, the creation of missing values is inevitable. For example, when the relation Animal is combined with the Female Parturition relation, missing values are created for every record which describes a cow less than two years of age since these cows are not old enough to produce calf.

The missing values created by joins will especially distort the data if the the missing values have meaning. In the previous example, if there is no value in the `parturition date' attribute of the Female Parturition relation, then this would mean that the cow did not have calf that year, whereas a missing value in cows under two years of age mean simply that the cow was too young to produce calf. The cow is a candidate for culling in the first instance while an acceptable cow in the second instance.

Embedded Meaning
There is embedded meaning within the structure of a relational database. One form of embedded meaning is the cardinality defining the type of relationship. For example, one cow (Animal relation) may give birth to many calves (New Born Animal relation). While these two relations are separated it is easy to see the type of relationship but when placed in one record, the cardinality is more difficult to determine.

Another form of embedded meaning is that the concept of time is shown in a relational structure, but is more difficult to represent in a two-dimensional flat file structure. In a relational database, one relation can represent data from one point in time and another can represent data from a different point in time. An example in the dataset was the Animal relation represented data that was recorded once in the life of a cow while the relation Test Day Production Detail was updated once a year. Once a record is combined from two time-different relations, it is difficult to perceive the change in time. It is difficult for humans to understand the data/time relationship when mixed up in a record, so we cannot expect the machine learning schemes to make sense of it.

Irrelevant Attributes
The requirement to link and join relations means that some attributes are duplicated in a relational database. The dataset needs to be carefully extracted to avoid these duplicated attributes which will only confuse the machine learning schemes.

The difficulties of creating a flat two-dimensional dataset from an efficient relational dataset structure will become more apparent as the project progresses. With the realisation that most databases created today are in relational form, comes the challenge for machine learning schemes to move away from the confines of a two-dimensional dataset to an SQL-based algorithm for data discovery.

5. Sources of Information

5.1 Livestock Improvement Corporation

Some additional information about the data was acquired from the Livestock Improvement Association Corporation. the interview notes shown in appendix C outline the additional information obtained from the Livestock Improvement Corporation.

5.2 Visiting Lecturer

A visiting lecturer experience in clustering gave some useful hints as to how to cope with the huge volume of data contained in the LIC dataset. Most importantly he highlighted the need to create new attributes which show the relationships between attributes and to consider time dependence of data. A diary note is shown in appendix D as a recording of the information gained by the lecturer's visit.

5.3 NZ Dairy Group

The New Zealand Dairy Board provided the following milk prices:

	$	
1986-87	3.60	per litre
1987-88	4.17	per litre
1989-89	5.78	per litre
1989-90	6.40	per litre
1990-91	4.43	per litre
1991-92	6.05	per kg milk fat
1992-93	6.50	per kg milk fat

The break-down for 1991-92 and 1992-93 is as follows:

	Fat (c)	Protein(c)	Milk Vol ($)
1991-92	298.54	528.355	-4.35
1992-93	319.76	572.42	-4.63

These milk prices were to be used to create an attribute which shows the monetary value of the cow's milk production.

6. Method of Research

The cow data was prepared for processing as follows:

1. LIC ran an extract on the original mainframe relational database to create the 60 files. The second set of 60 files (V262 data) to be merged with the main files was also created. The extracts were in COBOL format data ie. packed fields, no delimiters, field data format.

2. Using Unix tools (awk, grep & custom C programs), converted the COBOL extract files into ascii data delimited by commas. Merged the second extract file into the first extract files using a LIC index file. Broke each annual file of 705 field records up into eight files of 100 maximum fields to get around import restrictions of Lotus 123.

UNIX scripts were created to convert from raw data format to spreadsheet import format. This involved converting to a comma seperated form with text fields delimited with double quotes. Additionally, Lotus 123 required that import files to have 511 characters or less per line, so provision was made to split the data into files which fitted this requirement as well. Appendix E gives documentation for this UNIX script.

3. Imported all the data into Lotus 123 three dimensional spreadsheets. Initially Microsoft Access was used because of the relational nature of the LIC database, but this was later abandoned. A discussion of the Lotus 123 and Microsoft Access software tools in shown in appendix F.

4. Selected an initial subset of 24 attributes in an attempt to test the workbench and obtain some meaningful results quickly. The initial set of attributes were:

Animal Key
Animal Date of Birth
Transfer In Date (x3)
Transfer Out Date (x3)
Fate Code (x3)
Cause of Fate (x3)
Reproductive Status
Fat PI
Protein PI
Milk Volume PI
Payment PI
Fat BI
Protein BI
Milk Volume BI
Payment BI

For more information on these attributes refer to appendix A, the Description of Extract File from Livestock Improvement to get the page number for the LIC data dictionary entry.

5. Created a classification attribute which accurately represented the objective. (The WEKA_Status_Code classification is explained in detail in appendix G, Data Dictionary of WEKA Attributes)

6. Derived new attributes which could be interpreted meaningfully by the workbench. Some of the derived attributes are as follows:

Weka Age
Weka Bad Event
Weka X PI
Weka X BI
Weka Prev X PI
Weka Prev X BI
Weka XPI Change
Weka XBI Change
Weka AvgDiff X PI
Weka AvgDiff X BI

(X = fat, protein, milk volume, or payment)

For a detailed description of these attributes, appendix G contains a data dictionary of the WEKA attributes.

7. Exported an ASCII file from Lotus 123 to ARFF format and processed.

UNIX scripts were used to convert from a Lotus 123 print file to ARFF format for the workbench. Three different means were provided for converting exported data in slightly different formats. The first was for files with the titles in the first line of the file. The second was for files with the titles one on each line in seperate corresponding files. The last was for files with all the titles in one seperate file. These conversion scripts can handle whitespace or comma seperated data, with or without delimiting, and produce an ARFF file. Appendix H gives the documentation for these conversion scripts currently held in the

/export/local/ml/from.elsewhere/datasets/cow-culling/conversion

unix directory.

8. Iterated steps 4 through 6 with different selections of attributes, and additional derived attributes.

7. Results

7.1 C4.5 Output

C4.5 was the main scheme used throughout the project. The reasons for this were that C4.5

1. Could handle most facets of the data (real numbers, integers, discrete text values, missing values)

2. Produced easily interpreted results.

3. Ran quickly.

4. Ran consistently.

5. Output estimates of error.

6. Was capable of handling noise in the data.

7. Had a variety of options and parameters which meant that the induction process could be adjusted.

8. Was directly applicable to the problem domain, ie classifying cows to be culled or retained.

In particular, the ability of C4.5 to produce trees iteratively and to set a maximum number of instances per branch of the decision trees produced to avoid"over-fitting" to the data were very useful.

7.1.1 Initial Runs with Raw Data

When we first received our data from LIA, and had managed to convert some of it into ARFF format for the Weka Workbench, we had no domain knowledge. We decided to run this data through the workbench to see what resulted. Our first problem was in finding an attribute to use as a classification. The project had, from the beginning, been colloquially known as "The cow-culling project", and we were trying to classify cows as being culled or not culled. We initially looked to the fate codes as our classification, although we were somewhat confused by the lack of instances which had a value of "C", indicating that the cow had been culled, in this field.

Our first runs showed that the workbench definitely had the potential for classifying the cow-culling data, and also gave us some insight into the problems we would face. These runs produced trees and rules that would classify 85% of the cases correctly - however, these classifications turned out to be misleading. The first problem was that the classification attribute we were using (Fate Code) was incorrect for the actual classes we were looking for; this was a direct result of our unfamiliarity with the structure of the data. We later derived a field, Status Code, for classification.

The initial runs tended to split decision trees and produce clauses in rules based on irrelevant attributes. For example, there were several branches made on Animal Key, an Integer attribute containing a 9 digit unique identifier for the animal. Although there may be a partial ordering to these numbers (older animals tend to have lower Animal Keys, but not necessarily), this should have no bearing on classification. This highlighted the fact that the workbench schemes were susceptible to patterns in irrelevant attributes in the data.

Splits were also made on the field Animal Date Of Birth. In a run made on data from a single year, this would reflect the age of the animal. However, in a run made on data for several years, this would be lost, as the individual cases could be animals in different years (even the same animal), while the Animal Date Of Birth was absolute. We initially thought of putting in a field called Year, so that the workbench could produce splits such as;

```
Year = 1989
|    Animal_Date_Of_Birth < 840511 : Culled
```

However, the same problem would arise if the split was made on Animal Date Of Birth before year. We needed to derive a field for the animals age. We found that we would have to scrutinize attributes to see if they really were relevant to classification, or whether they were not. We also had to derive some implied attributes which were relevant to the classification, but at the same time being careful not to destroy the original meaning of the data, or create entirely spurious new attributes.

Some of the results using the WXV data only from the C4.5 scheme contained on the WEKA workbench are as follows:

The following is part of a decision tree produced by C4.5, which illustrates some of the problems we had with the data. This decision tree was made using FateCode as the classification. FateCode has several possible values. In this dataset, the values were S (Sold), D (Dead), L (lost) and G (Gone - unaccounted for).

```
Simplified Decision Tree:

Transferoutdate <= 900420 :
|   Transferoutdate <= 880217 : G (10.0/2.4)
|   Transferoutdate > 880217 :
|   |   AnimalDateofBirth > 860811 : D (7.0/2.4)
|   |   AnimalDateofBirth <= 860811 :
|   |   |   Transferoutdate <= 890613 :
|   |   |   |   Causeoffate = 1A: S (0.0)
|   |   |   |   Causeoffate = BL: D (5.0/1.2)
|   |   |   |   Causeoffate = CT: S (0.0)
|   |   |   |   Causeoffate = GS: S (0.0)
|   |   |   |   Causeoffate = IN: S (0.0)
|   |   |   |   Causeoffate = LP: S (4.0/1.2)
|   |   |   |   Causeoffate = MF: S (0.0)
|   |   |   |   Causeoffate = MT: S (9.0/1.3)
|   |   |   |   Causeoffate = OA: S (0.0)
|   |   |   |   Causeoffate = UD: S (0.0)
|   |   |   |   Causeoffate = OC:
|   |   |   |   |   MatingDate <= 861220 : S (74.0/5.0)
|   |   |   |   |   MatingDate > 861220 :
|   |   |   |   |   |   AnimalKey <= 2.81151e+06 : D (2.0/1.0)
|   |   |   |   |   |   AnimalKey > 2.81151e+06 : S (4.0/1.2)
```

The first split which is made in this tree is on Transferoutdate. This was undesirable for several reasons. First, splits made on Transfer out dates are useless for the rules we want to generate. It is essentially saying "If you removed the cow from the herd then you removed the cow from the herd". This illustrates that in datasets such as this, C4.5 could be used to find out what happened historically. However, we are not trying to reconstruct past events, but predict future ones. Secondly, in the dataset, there were three fields named Transferoutdate. Until these were renamed, we didn't know which one of these fields was being split on. These three fields together had meaning, but on their own they were ambiguous. Third, there were many missing values in these fields, and missing values actually meant that the cow had not been transferred out of the herd, which misled C4.5.

Splits are also made very early on AnimalDateofBirth. If all the data were for the same year, this could be usable as a measure of the cow's age, but in data covering 6 years, absolute date values have little meaning, remembering that we are trying to derive general rules, rather than create an historical model of culling patterns.

Splits on Causeoffate again reflect the structure of the data more than decisions to determine whether an animal should be culled. Every instance with a value for FateCode has a value for Causeoffate (bar a few missing values). The problem with the fate code is that it does not differentiate between those cows which died due to some random event such as milk fever, and those who were removed from the herd due to the farmer's decision.

For example, the leaf node which says "if Causeoffate = LP then class = S" means "If the cow is a low producer then sell it". This is a logical rule, and in fact, one of the main reasons why animals are removed from milking herds. However, it does not say anything about how to determine if the animal is a low producer. The Causeoffate LP is a code meaning that the farmer has identified that cow as a low producer, and we wanted the schemes to learn how to identify low producing cows from production data.

The splits on animal key are irrelevant. Although AnimalKey is an Integer field, it only serves as a unique identifier by which each cow may be referenced. Although there may be a partial ordering as younger animals get higher values, making culling decisions based on the value of the cow's identification number is ridiculous.

Looking at the correlation matrix for the corresponding ruleset:

```
        Tested 1422, errors 74 (5.2%)   <<

                  (a)        (b)     (c)     (d)        <-classified as
                  ----        ----   ----     ----
                  214        17              10         (a): class D
                  11         759             10         (b): class G
                  2                                     (c): class L
                  8          16              375        (d): class S
```

we see an error rate of only 5.2%. The results look good, and it appears from the matrix that the rules are good, bar some noise in the data. However, most of the rules are of no practical use, even though they accurately classify most of the data.

After we realised that the data in raw form was not suitable for deriving the kind of rules and classifications we wanted form the workbench, we set about trying to get it into a form which was. We had to be very careful not to destroy the meaning of the original data, or to create entirely spurious new fields.

What we learned from these runs

Although the early runs did not produce the culling rules we were looking for, they did provide us with some insight into the structure of the data, correlations between attributes, and which attributes would have to be altered in order to produce results which were useful.

Because the datasets were going to contain data from many different years, absolute date values would have to be replaced with relative date values. In particular, the Date of Birth of the cow was replaced with the age of the cow.

These runs also showed the high correlation between fate code and cause of fate. The early runs showed up very quickly which attributes were highly correlated and dependent on each other.

7.1.2 Subsequent runs with modified data

Tree Produced By C4.5 Decision Tree Generator
This is the tree output form C4.5 from our first run with our lotus-munged data. Only 11 attributes were used for this run, but it produced some very good rules, and incorrectly classifies only 4.7% of the cases. Previous runs with partially munged data typically had errors of 12-15%.

```
Read 2686 cases (11 attributes) from /tmp/arffdoc45.4720.data
```

Decision Tree:

```
WEKA_Age <= 10 :
|   WEKA_Age <= 2 : retained (698.6/11.3)
|   WEKA_Age > 2 :
|   |   WEKA_FatPl_DiffAvg <= -18 :
|   |   |   WEKA_MilkVolumePl_DiffAvg <= -34.2 :
|   |   |   |   WEKA_PaymentPl_DiffAvg > -31.72 : retained (3.7/1.1)
|   |   |   |   WEKA_PaymentPl_DiffAvg <= -31.72 :
|   |   |   |   |   WEKA_Age > 6 : departed (5.7/1.5)
|   |   |   |   |   WEKA_Age <= 6 :
|   |   |   |   |   |   WEKA_Age <= 3 : departed (4.2/1.2)
|   |   |   |   |   |   WEKA_Age > 3 : retained (3.2/0.2)
|   |   |   WEKA_MilkVolumePl_DiffAvg > -34.2 :
|   |   |   |   WEKA_PaymentBl_DiffAvg <= -4.2 :
|   |   |   |   |   WEKA_ProteinBl_DiffAvg <= -14.91 : retained (12.5/3.4)
|   |   |   |   |   WEKA_ProteinBl_DiffAvg > -14.91 :
|   |   |   |   |   |   WEKA_FatBl_DiffAvg <= -14.17 : retained (46.8/1.6)
|   |   |   |   |   |   WEKA_FatBl_DiffAvg > -14.17 :
|   |   |   |   |   |   |   WEKA_FatPl_DiffAvg <= -18.34 :
|   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg > -6.06 : retained (29.9/1.1)
|   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg <= -6.06 :
|   |   |   |   |   |   |   |   |   WEKA_PrtnBl_DiffAvg <= -11.08 : retained (16.8/0.6)
|   |   |   |   |   |   |   |   |   WEKA_ProteinBl_DiffAvg > -11.08 :
|   |   |   |   |   |   |   |   |   |   WEKA_ProteinBl_DiffAvg <= -10.21 :[S1]
|   |   |   |   |   |   |   |   |   |   WEKA_ProteinBl_DiffAvg > -10.21 :[S2]
|   |   |   |   |   |   |   WEKA_FatPl_DiffAvg > -18.34 :
|   |   |   |   |   |   |   |   WEKA_Age <= 7 : retained (6.3/0.3)
|   |   |   |   |   |   |   |   WEKA_Age > 7 :
|   |   |   |   |   |   |   |   |   WEKA_Age <= 8 : departed (2.3/0.3)
|   |   |   |   |   |   |   |   |   WEKA_Age > 8 : retained (2.5/1.1)
|   |   |   |   WEKA_PaymentBl_DiffAvg > -4.2 :
|   |   |   |   |   WEKA_Calved = Y: retained (2.9/0.0)
|   |   |   |   |   WEKA_Calved = N:
|   |   |   |   |   |   WEKA_ProteinPl_DiffAvg <= -10.61 : retained (4.1/1.2)
|   |   |   |   |   |   WEKA_ProteinPl_DiffAvg > -10.61 :
|   |   |   |   |   |   |   WEKA_Age <= 3 : departed (2.7/0.7)
|   |   |   |   |   |   |   WEKA_Age > 3 : retained (3.4/1.2)
|   |   WEKA_FatPl_DiffAvg > -18 :
|   |   |   WEKA_Calved = Y: retained (279.4/2.8)
|   |   |   WEKA_Calved = N:
|   |   |   |   WEKA_MilkVolumePl_DiffAvg > 12.9 : retained (259.6/12.0)
|   |   |   |   WEKA_MilkVolumePl_DiffAvg <= 12.9 :
|   |   |   |   |   AvgDiff_MV_Bl <= 7.52 :
|   |   |   |   |   |   WEKA_FatPl_DiffAvg <= -3.64 :
|   |   |   |   |   |   |   WEKA_FatPl_DiffAvg <= -14.24 : retained (81.4/3.4)
|   |   |   |   |   |   |   WEKA_FatPl_DiffAvg > -14.24 :
|   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg <= -1.6 :
|   |   |   |   |   |   |   |   |   WEKA_FatBl_DiffAvg > -9.37 : retained (158.8/8.7)
|   |   |   |   |   |   |   |   |   WEKA_FatBl_DiffAvg <= -9.37 :
|   |   |   |   |   |   |   |   |   |   WEKA_Age <= 8 : retained (9.1/0.9)
|   |   |   |   |   |   |   |   |   |   WEKA_Age > 8 : departed (3.8/1.5)
|   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg > -1.6 :
|   |   |   |   |   |   |   |   |   WEKA_PaymentPl_DiffAvg > -0.75 : retained (23.2/1.0)
|   |   |   |   |   |   |   |   |   WEKA_PaymentPl_DiffAvg <= -0.75 :
|   |   |   |   |   |   |   |   |   |   WEKA_ProteinPl_DiffAvg <= 4.69 :[S3]
|   |   |   |   |   |   |   |   |   |   WEKA_ProteinPl_DiffAvg > 4.69 :[S4]
|   |   |   |   |   |   WEKA_FatPl_DiffAvg > -3.64 :
|   |   |   |   |   |   |   WEKA_Age <= 6 :
|   |   |   |   |   |   |   |   WEKA_MilkVolumeBl_DiffAvg <= 3.61 : retained (257.6/9.1)
|   |   |   |   |   |   |   |   WEKA_MilkVolumeBl_DiffAvg > 3.61 :
|   |   |   |   |   |   |   |   |   WEKA_MilkVolumeBl_DiffAvg > 3.81 : retained (115.8/4.7)
|   |   |   |   |   |   |   |   |   WEKA_MilkVolumeBl_DiffAvg <= 3.81 :
|   |   |   |   |   |   |   |   |   |   WEKA_FatPl_DiffAvg <= 8.16 : departed (4.0/1.9)
|   |   |   |   |   |   |   |   |   |   WEKA_FatPl_DiffAvg > 8.16 : retained (6.0/1.2)
|   |   |   |   |   |   |   WEKA_Age > 6 :
|   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg <= 3.8 : retained (105.8/7.5)
|   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg > 3.8 :
|   |   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg > 4.61 : retained (30.5/3.2)
|   |   |   |   |   |   |   |   |   WEKA_PaymentBl_DiffAvg <= 4.61 :
|   |   |   |   |   |   |   |   |   |   WEKA_Age <= 7 : departed (4.2/2.0)
|   |   |   |   |   |   |   |   |   |   WEKA_Age > 7 : retained (3.0/0.3)
|   |   |   |   |   WEKA_MilkVolumeBl_DiffAvg > 7.52 :
|   |   |   |   |   |   WEKA_MilkVolumeBl_DiffAvg <= 7.62 : departed (3.9/1.7)
|   |   |   |   |   |   WEKA_MilkVolumeBl_DiffAvg > 7.62 :
|   |   |   |   |   |   |   WEKA_FatBl_DiffAvg <= 13.63 :
|   |   |   |   |   |   |   |   WEKA_FatPl_DiffAvg > 6.26 : retained (38.7/1.6)
|   |   |   |   |   |   |   |   WEKA_FatPl_DiffAvg <= 6.26 :
|   |   |   |   |   |   |   |   |   WEKA_MilkVolumePl_DiffAvg <= 6.4 : retained (5.8/0.2)
|   |   |   |   |   |   |   |   |   WEKA_MilkVolumePl_DiffAvg > 6.4 :
|   |   |   |   |   |   |   |   |   |   WEKA_Age <= 4 : departed (3.4/1.4)
|   |   |   |   |   |   |   |   |   |   WEKA_Age > 4 : retained (2.4/0.2)
|   |   |   |   |   |   |   WEKA_FatBl_DiffAvg > 13.63 :
|   |   |   |   |   |   |   |   WEKA_Age <= 3 : departed (3.7/1.6)
|   |   |   |   |   |   |   |   WEKA_Age > 3 : retained (2.1/0.2)
WEKA_Age > 10 :
|   WEKA_ProteinBl_DiffAvg <= -11.31 :
```

```
|   |   WEKA_FatPl_DiffAvg <= -22.31 : retained (3.9/1.5)
|   |   WEKA_FatPl_DiffAvg > -22.31 :
|   |   |   WEKA_Age > 15 : departed (3.1)
|   |   |   WEKA_Age <= 15 :
|   |   |   |   WEKA_Age <= 11 : departed (3.0/0.6)
|   |   |   |   WEKA_Age > 11 : retained (3.5/0.7)
|   WEKA_ProteinBl_DiffAvg > -11.31 :
|   |   WEKA_PaymentPl_DiffAvg > 9.19 : retained (55.0/5.5)
|   |   WEKA_PaymentPl_DiffAvg <= 9.19 :
|   |   |   WEKA_ProteinPl_DiffAvg <= 8.82 :
|   |   |   |   WEKA_FatPl_DiffAvg > -0.81 : retained (33.0/3.3)
|   |   |   |   WEKA_FatPl_DiffAvg <= -0.81 :
|   |   |   |   |   WEKA_FatBl_DiffAvg <= -5.33 :
|   |   |   |   |   |   WEKA_FatBl_DiffAvg > -11.63 : retained (58.7/6.8)
|   |   |   |   |   |   WEKA_FatBl_DiffAvg <= -11.63 :
|   |   |   |   |   |   |   WEKA_MilkVolumePl_DiffAvg <= -22.6 : departed (3.7/1.3)
|   |   |   |   |   |   |   WEKA_MilkVolumePl_DiffAvg > -22.6 :
|   |   |   |   |   |   |   |   WEKA_MilkVolumePl_DiffAvg <= -13.16 : retained (11.0/1.1)
|   |   |   |   |   |   |   |   WEKA_MilkVolumePl_DiffAvg > -13.16 :
|   |   |   |   |   |   |   |   |   WEKA_Calved = N: retained (2.6/0.5)
|   |   |   |   |   |   |   |   |   WEKA_Calved = Y: departed (2.9/0.8)
|   |   |   |   |   WEKA_FatBl_DiffAvg > -5.33 :
|   |   |   |   |   |   WEKA_ProteinPl_DiffAvg <= -8.68 : departed (3.7/1.3)
|   |   |   |   |   |   WEKA_ProteinPl_DiffAvg > -8.68 :
|   |   |   |   |   |   |   WEKA_Age > 12 : retained (2.1/0.3)
|   |   |   |   |   |   |   WEKA_Age <= 12 :
|   |   |   |   |   |   |   |   WEKA_ProteinBl_DiffAvg <= -3.18 : retained (2.6/0.2)
|   |   |   |   |   |   |   |   WEKA_ProteinBl_DiffAvg > -3.18 : departed (2.6/0.4)
|   |   |   WEKA_ProteinPl_DiffAvg > 8.82 :
|   |   |   |   WEKA_FatBl_DiffAvg <= -2.37 : retained (5.5/0.5)
|   |   |   |   WEKA_FatBl_DiffAvg > -2.37 :
|   |   |   |   |   WEKA_Age <= 11 : departed (2.5/0.3)
|   |   |   |   |   WEKA_Age > 11 : retained (3.0/1.3)
```

Subtree [S1]

```
WEKA_PaymentPl_DiffAvg > -23.95 : retained (5.6/0.2)
WEKA_PaymentPl_DiffAvg <= -23.95 :
|   WEKA_MilkVolumePl_DiffAvg <= -28.51 : retained (3.7/0.1)
|   WEKA_MilkVolumePl_DiffAvg > -28.51 :
| |  |   WEKA_Calved = N: retained (4.9/2.3)
|   |   WEKA_Calved = Y: departed (2.5/0.5)
```

Subtree [S2]

```
WEKA_PaymentBl_DiffAvg <= -9.39 : retained (26.2/0.9)
WEKA_PaymentBl_DiffAvg > -9.39 :
|   22WEKA_Age <= 4 : retained (31.5/2.0)
|   WEKA_Age > 4 :
|   |   WEKA_MilkVolumeBl_DiffAvg <= -7.76 :
|   |   |   WEKA_FatPl_DiffAvg > -19.41 : retained (5.0/0.2)
|   |   |   WEKA_FatPl_DiffAvg <= -19.41 :
|   |   |   |   WEKA_FatBl_DiffAvg > -8.13 : retained (3.3/0.1)
|   |   |   |   WEKA_FatBl_DiffAvg <= -8.13 :
|   |   |   |   |   WEKA_Age <= 6 : retained (2.3/1.1)
|   |   |   |   |   WEKA_Age > 6 : departed (4.3/1.2)
|   |   WEKA_MilkVolumeBl_DiffAvg > -7.76 :
|   |   |   WEKA_PaymentPl_DiffAvg > -23.32 : retained (33.1/2.2)
|   |   |   WEKA_PaymentPl_DiffAvg <= -23.32 :
|   |   |   |   WEKA_ProteinPl_DiffAvg <= -16.41 : departed (3.3/1.2)
|   |   |   |   WEKA_ProteinPl_DiffAvg > -16.41 : retained (3.3/0.1)
```

Subtree [S3]

```
WEKA_PaymentBl_DiffAvg > 0.3 : retained (48.4/3.1)
22 WEKA_PaymentBl_DiffAvg <= 0.3 :
|   WEKA_PaymentBl_DiffAvg <= 0.01 :
|   |   WEKA_FatPl_DiffAvg > -9.21 : retained (23.2/1.0)
|   |   22 WEKA_FatPl_DiffAvg <= -9.21 :
|   |   |   WEKA_ProteinBl_DiffAvg <= -0.58 : departed (5.8/2.6)
|   |   |   WEKA_ProteinBl_DiffAvg > -0.58 : retained (7.7/0.3)
|   WEKA_PaymentBl_DiffAvg > 0.01 :
|   |  2 WEKA_Age <= 4 : departed (4.9/1.8)
|   |   WEKA_Age > 4 : retained (2.8/0.2)
```

Subtree [S4]

```
WEKA_FatBl_DiffAvg <= -4.49 : retained (3.9/0.2)
WEKA_FatBl_DiffAvg > -4.49 :
|   WEKA_PaymentPl_DiffAvg <= -2.91 : retained (3.9/1.2)
|   WEKA_PaymentPl_DiffAvg > -2.91 :
|   |   WEKA_Age <= 3 : departed (2.7/0.6)
|   |   WEKA_Age > 3 : retained (3.1/1.2)
```

Simplified Decision Tree From C4.5
This is a simplification of the above tree. Actually, it's more of an over-simplification. Because C4.5 considers 6.5% error is acceptable, and only 6.5% of the cases have class departed, the simplified tree consists of one leaf node containing the class retained.

```
Simplified Decision Tree:
 retained (2686.0/184.6)

Evaluation on training data (2686 items):

        Before Pruning              After Pruning
        Size          Errors              Size    Errors              Estimate
        157       125( 4.7%)        1               175( 6.5%)           ( 6.9%)
```

Rules Produced By C4.5 Rule Generator
Final rules from tree 0:

```
Rule 52:
        WEKA_Age > 10
        WEKA_FatPI_DiffAvg > -22.31
        WEKA_ProteinBI_DiffAvg <= -11.31
        ->  class departed  [75.8%]
```

If a cow is over 10 years old , and has a protein BI which is below the herd average by more than 11.31, and a fat PI which is not less than 22.31 below the herd average, then the cow leaves the herd.

The part of the rule referring to fat PI being above a certain value is misleading, and is possibly caused by the greedy information gain heuristic used by C4.5. This rule correctly classifies 6 of the 7 cases to which it applies. However, upon examination of the data, we found that modifying this rule to;

```
        WEKA_Age >10
        WEKA_FatPI_DiffAvg < -2.23
        WEKA_ProteinBI_DiffAvg <= -11.31
        ->  class departed
```

would correctly classify 7 out of 9 cases.

```
Rule 2:
        WEKA_MilkVolumePI_DiffAvg <= -34.2
        ->  class departed  [73.1%]
```

If a cow has a Milk Volume PI which is below the herd average by more than 34.2 then the cow leaves the herd. This indicates that cows are sold or culled due to low production. This rule correctly classifies 8 out of 11 cases.

```
Rule 27:
        WEKA_Age <= 4
        WEKA_PaymentBI_DiffAvg > 0.01
        WEKA_PaymentBI_DiffAvg <= 0.3
        ->  class departed  [63.0%]
```

This rule correctly classifies 3 out of 3 cases, but it's a pretty hairy rule which, once again, is probably caused by the greedy algorithm used by C4.5 being influenced by noise in the data. We haven't been able to find a reasonable alternative explanation for this rule.

```
Rule 62:
        WEKA_PaymentPI_DiffAvg > 9.19
        ->  class retained  [97.1%]
```

If a cow has a Payment PI greater than the herd average by more than 9.19, then the cow is retained. This rule correctly classifies 342 out of 350 cases.

```
Rule 3:
        WEKA_Age <= 6
        -> class retained  [95.2%]
```

If a cow is under 6 years old, it is retained. This rule correctly classifies 1911 out of 2004 cases. Of course, the cow has to a\have slipped through all the above rules to make it this far...

```
Rule 10:
        WEKA_Age <= 10
        WEKA_MilkVolumePI_DiffAvg > -34.2
        WEKA_ProteinBI_DiffAvg > -10.21
        -> class retained  [94.2%]
```

If a cow is under 10 years old, it's Milk Volume PI is better than 34.2 below average and it's Protein BI is better than -10.21 below average, then the cow is retained.

Default class: retained

Anything which slips through all the above rules without receiving a clasification is assigned the default class retained.

Evaluation and Errors
Evaluation on training data (2686 items):

```
Rule    Size    Error     Used     Wrong      Advantage
 52     3       24.2%     5        0 (0.0%)     5 (5|0) departed
  2     1       26.9%     9        1 (11.1%)    7 (8|1)        departed
 27     3       37.0%     3        0 (0.0%)     3 (3|0)        departed
 62     1        2.9%   292        6 (2.1%)     0 (0|0)        retained
  3     1        4.8%  1723       75 (4.4%)     0 (0|0)        retained
 10     3        5.8%   233       10 (4.3%)     0 (0|0)        retained
```

The highest percentage of cases wrongly classified by any of the rules generated is 4.4%, and the total error is 6.0%.

```
Tested 2686, errors 160 (6.0%)   <<

                C4.5 CLASSIFICATION
                departed       retained        ACTUAL CLASSIFICATION
                16             159             class departed
                1              2510            class retained
```

Here, we see that of 2511 cases with a class of retained, 2510 were correctly classified. Hooray!
BUT...of 175 cases with a class of departed, only 16 were classified correctly! That's only 9.1% accuracy...hardly inspiring! However, there is good news! The status code attribute, which was used as the classification in these runs, was based on the transfer in and out dates of the animal, not on what caused it's departure from the herd. So, a perfectly healthy cow with good PI's and BI's could be struck by lightning, causing it to leave the herd and giving it a status code of departed, but would lead C4.5 to class it as retained. We are currently testing this hypothesis...

7.1.3 More impressive results:

Once we had processed the data sufficiently, we made many more runs through the workbench. The fields which we had modified or derived were given the prefix `WEKA_'. Following is output from C4.5. The classification for this tree was Weka_Status_Code. This has three possible values: retained, indicating that the cow was retained in the herd, dep_random, indicating that the cow departed from the herd due to a random event, and dep_farmer, indicating that the farmer decided to remove this animal from the herd.

```
Simplified Decision Tree:

WEKA_Bad_Event = Y: dep_random (19.0/0.0)
WEKA_Bad_Event = N:
|   WEKA_Age <= 2 : retained (805.1/10.4)
|   WEKA_Age > 2 :
|   |   WEKA_PaymentBI_DiffAvg > -10.8 : retained (1847.9/109.8)
|   |   WEKA_PaymentBI_DiffAvg <= -10.8 :
|   |   |   WEKA_MilkVolumePI_DiffAvg <= -33.93 : dep_farmer (10.5/3.8)
|   |   |   WEKA_MilkVolumePI_DiffAvg > -33.93 : retained (84.5/15.2)
```

This is the complete simplified decision tree produced by C4.5. This is much more concise than trees produced by the previous runs. The first split is made on the field WEKA_Bad_Event. This field indicates whether a random, disastrous event (such as a fatal injury or disease) has struck the cow, causing it to be removed from the herd. This field was derived from the fate code and cause of fate fields, and was included to differentiate between animals which had left the herd due to random causes and those which had left the herd due to the farmer's decision.

The next split is made on WEKA_Age. This is simply the age of the cow (to the nearest year), and is derived from the AnimalDateofBirth field and the year of the extract data. Note that the tree states that animals 2 years old or under are retained. We learned later that these animals, because they have not yet begun lactation, are subject to different culling rules than older animals. This branch of the decision tree shows that C4.5 has learned this. We decided also to split the data into those animals 2 years old and under and those over two years old, and perform runs on the sets separately as well.

The next splits are made on WEKA_PaymentBI_DiffAvg and WEKA_MilkVolumePI_DiffAvg. These fields are the linear deviation of the animal's Payment BI and Milk Volume PI scores from the herd averages. We see that if the Payment BI (estimate of the expected return from the cow during it's lifetime) is better than 10.8 below the herd average then the animal is retained. If not, then we look at the milk volume PI(measure of actual milk volume produced). If this is worse than 33.93 below the herd average, then the farmer removes the animal from the herd. The combination of these two branches reflects the culling of low-producing cows.

The rules produced by the same run are;

```
Composite ruleset:

Rule 1:
        WEKA_Bad_Event = Y
        ->  class dep_random  [93.0%]

Rule 22:
        WEKA_Prtn_BI > 118.4
        WEKA_ProteinBI_DiffAvg <= -0.38
        WEKA_Fat_PI_Change > -0.2
        WEKA_Fat_PI_Change <= 0
        ->  class dep_farmer  [79.4%]

Rule 13:
        WEKA_MilkVolumePI_DiffAvg <= -33.93
        WEKA_PaymentPI_DiffAvg <= -24.91
```

```
                -> class dep_farmer [64.5%]

        Rule 7:
                WEKA_Age > 10
                WEKA_PaymentBI_DiffAvg <= -11.49
                -> class dep_farmer [44.2%]

        Rule 4:
                WEKA_Bad_Event = N
                -> class retained [94.6%]

        Default class: retained
```

Rule 1 classifies all those cows which have left the herd due to a random event.

Rule 22 is ambiguous. This rule states that if there has been very little change in the animals Fat PI score from the previous year to this year (between -0.2 and 0), and it's Protein BI score is greater than 118.4, but it is below the herd average Protein BI score by more than 0.38, then the farmer removes the cow from the herd.

Rule 13 culls out low producing cows. The rule says that if the animal's Milk Volume PI is below the herd average by -33.93 or more and it's Payment PI is below the herd average by 24.91 or more, then the farmer removes it from the herd.

Rule 7 culls out the older cows which are falling behind in production. If the cow is more than 10 years old and is below the herd average Payment BI by more than 11.49, then the farmer removes it from the herd.

Rule 4 says that any other cow is retained in the herd, unless some bad event happens to it (in which case it would have been trapped by rule 1).

```
Tested 2767, errors 127 (4.6%)   <<

            (a)      (b)      (c)           <-classified as
            ----     ----     ----
             21               120        (a): class dep_farmer
                      19                 (b): class dep_random
              7               2600       (c): class retained
```

The correlation matrix for the run indicates only 4.6% error. The main error seems to be over-classifying retained animals. This is probably because not enough information has been given to C4.5 to distinguish between retained cows and those removed by the farmer. Some good rules have been produced for determining whether the farmer should remove a cow, but more attributes need to be given to the workbench before the mis-classified cases can be correctly classified.

7.2 Output from Other Schemes

As well as C4.5, there are 8 other schemes on the workbench, and all of these were used on the cow-culling dataset. The most productive of these was FOIL, 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.

FOIL provided a slightly different set of rules to those produce by C4.5, because the rules it produced were for only one classification at a time, where the ripple down rules produced by C4.5 provided rules for all the available classifications.

FOIL had the tendency to overfit the data when the number of instances left to produce rules from was small, but the results were consistent with those produced by C4.5, which was encouraging. The only slight problem with FOIL was interpreting the results, which did not include the attribute names, which had to be cross-referenced.

The pFOIL schemes, CNF and DNF were limited in their usefulness because of their inability to handle continuous or ordered data, but Implementation problems with Prism meant that it always crashed or had to be killed, so no useful results were gained there.

Our implementation of Induct, the more capable descendant of Prism, was limited by it's inability to handle numeric attributes. Methods were provided to quantise numeric data into discrete ranges, but with no concept of ordering between the ranges, results from Induct were not as useful as the results from other schemes.

At various stages throughout the project, attempts were made to run the data through the two clustering schemes on the workbench. Both Cobweb and Autoclass are unsupervised learning schemes which attempt to discover classes within a dataset.

Unfortunately, the runs through Autoclass proved to be divergent, where Autoclass found more classes the longer it was left to run. We had hoped that Autoclass would discover classes similar to the "retained" and "departed" classes we had defined, but this was not the case. This is an indication that there was not enough information present in the data for Autoclass to be able to discover a small number of well defined classes.

Cobweb had produced good results with data from a single participant for a single year, but the dataset for 6 years had approximately 3000 records, and this caused Cobweb to run out of memory and disk space due to the large number of temporary files used.

OC1 interprets the data as a hyperspace with one dimension for each attribute, and produces rules for classification in the form of hyperplanes dividing the space into distinct regions. This can be used to produce a graph, if the data is 2 dimensional with a classification attribute, but when the data goes beyond 2 dimensions, it is very difficult to visualise the results. The data we were using had a minimum of 20 dimensions, so we did not use OC1 very much.

7.3 Data Manipulation

Considerable effort was spent on converting the raw data into linear form in preparation for the WEKA workbench. The majority of the data manipulation was performed using Lotus 123 for DOS (version 3.4). Initially Microsoft Access (version 1.1) was used but this was abandoned. Using these packages, some considerable time was spent converting from their ASCII export formats to the WEKA workbench format.

The hardware used with Lotus 123 and MS Access consisted of an IBM compatible, 386 SX with five megabytes of RAM and a 500 meg disk drive. Swap drives were used to create additional virtual (slow) RAM. If Lotus 123 or MS Access are to be used again for data manipulation of such large quantities of data, then a computer with a math co-processor and 16 mb of RAM would be the minimum hardware recommendation. (Lotus 123 for unix would be the ideal package but, dreams are free!)

Following is an analysis of the advantages and disadvantages of these software packages and also a general list of data manipulation requirements.

7.3.1 Advantages/Disadvantages of Microsoft Access

Microsoft Access version 1.1 was used in conjuction with Window version 3.1.

Advantages

Can handle data presented in a relational format.

Could import similarly structured datasets in raw COBOL format once an Import/Export Specification is correctly designed.

Query facility to test results of the workbench schemes.

ASCII export feature allowed the first record to contain the attributes names. This made conversion of subsets of attributes to WEKA format more flexible.

The ability to view the data in row by column format.

Disadvantages

Microsoft Access is full of bugs which caused major delays and were often difficult to work around. These bugs included:

Crashing to the DOS prompt when the Windows swap drive was installed.

The query facility on calculations e.g. average would not use the criteria option available.

Access could not import dates in YYMMDD format.

Other disadvantages included:

The setting up of the import/export specification required exact knowledge of the structure including which column each of 705 attributes started in.

You can not create macros to perform repetitive operations on table in a similar format. Since we had 60 tables in a similar format this was a major drawback.

Microsoft Access would not accept question marks as missing values so it was difficult to determine what was a missing value and what was supposed to have a value of zero.

Only 259 columns can be addressed in on table.

Slow to use with large (139 column) tables.

7.3.2 Advantages/Disadvantages of Lotus 123

Advantages

The three dimensional spreadsheet feature allowed all six years for one participant to be operated on simultaneously. This feature was expecially useful for adding, deleting and creating new attributes.

The ability to copy the data from the record of a cow in one year to the record of the same cow in another year. Used to obtain the WEKA Previous PI & BI figures.

Query facility to gain a better understanding of the data and to check the workbench output.

The ability to view the data for improved understanding.

An extremely flexible query facility which can use criteria.

Forumulas could be used to create new attributes and these were easily modified to perfection.

Graphs were simple and quick to generate to gain further understanding of the data.

Disadvantages

Lotus 123 was slow to manipulate the columns given the hardware constraints.

Import restrictions to 512 characters meant breaking up the 705 attributes into 100 attributes files first, and only comma-separated format was acceptable.

259 columns limitation.

The only method of export to ASCII format is via a print file which later needed to be comma separated.

7.3.3 General Data Manipulation Requirements

To be able to derive new attributes from existing attributes using formulae.

To be able to view the data to ensure the changes made to attributes are correct. Viewing the data helps determine trends which contribute to a better understanding of the data.

The ability to perform queries to test the hypotheses and rules produced by the workbench schemes and to analyse the contents of attributes for a better understanding.

An easy method of extracting subsets of attributes.

The ability to create graphs of each attribute or comparisions of each attribute. e.g. line graphs for trends, histograms for frequency distributions (currently available on the workbench).

Statistical function e.g average, standard deviation.

IF THEN ELSE formulae for deriving attributes.

7.4 Submission of Paper

A paper showing the results of this project is shown in Appendix I

8. Future Direction

Contained in the UNIX lucifer directory /home/ml/datasets/cow_culling are 10 datasets each containing six years of data for one participant with the initial 705 attributes and the WEKA attributes. The datasets are in the format required by the WEKA workbench (i.e. ARFF format).

To come from the initial runs which produced useless trees and rules to the latest runs which classified more than 95% of the cases correctly and produced usable trees and rules required extensive alterations and derivations of the the original data.

The latest runs are a long way from the initial runs, but there is still a long way to go, as the latest runs still incorrectly classify a large proportion of animals whihc the farmer has removed from the herd. There are still a large number of attributes in the original data which ahve not yet been modified and used in the workbench runs since the initial runs. These will probably help correctly classify most of the incorrectly classified animals, once they are in a form which the workbench can use.

For those who want to continue grapling with this cow-culling dataset we suggest the following approach:

Classification

Separate the dep_farmer classification further by cause of fate. For example have a dep_farmer_LP for an animal which has permanently left the herd from a decision made by the farmer about Low Production. You may identify just what production level a farmer considers to be low production.

Data

Try existing datasets other than WXV.

Try existing datasets with all records from 10 participants in in one file.

One way to tackle large datasets is to start with a dataset containing the selected attributes and the new WEKA attributes and gradually build onto this subset getting to know the data thoroughly as you go. NOTE: the objective is to eventually get all the relevant attributes from the 705 through the workbench in a form that the workbench will gains maximum knowledge.

Data Manipulation

Remembering that none of the selected attributes were used in their raw form to get results from the schemes, consider each attribute individually and determine the best linear format for the scheme you are to use. Start with C4.5 since this has produced the best results so far.

New data

Encorporate the milk price into an attribute to give the monetary value of the animal's production. Refer appendix C LIC interview notes.

Objective

Change the objective to move away from a culling decision made by a farmer to the decision to retain a cow or some other subjective decision which may be represented in the data structure.

APPENDIX A

Description of Extract File from Livestock Improvement

Animal

Y001CTRL	9(9)	N	9	Animal Key	1

Animal Birth Identification

VAIDPOWN	X(4)	A	4	Participant (owner) code	243
VAIDYRID	9(4)	N	4	Year identified	243
VAIDNAOY	9(4)	N	4	????	

Animal

Y001SIRE	9(9)	N	9	Animal Sire	1

AB or Grade Bull

VAIDBULL	9(6)	N	6	????	

Animal

Y001ACCP	9(1)	N	1	Sire Official Code	1
Y001DAMM	9(9)	N	9	Animal Dam	1
Y001BRTH	9(6)	N	6	Animal Date of Birth	2
Y001DTHC	9(1)	N	1	DOB estimated?	2
Y001SEXX	X(1)	A	1	Animal Sex	2

Test Number Identification

VLNMNUMA	9(5)	N	5	Test number	136

Animal Location (3 times)

VLNMLOCN	X(10)	A	10	Location of Farm	22
VLNMHERD	9(2)	N	2	Herd Number in Farm	22
VLNMIDAT	9(6)	N	6	Transfer in date	22
VLNMODAT	9(6)	N	6	Transfer out date	22
VLNMFATC	X(1)	A	1	Fate code	22
VLNMCCFC	X(2)	A	2	Cause of fate	23

Female Parturition

VBIRPDAT	9(6)	N	6	Parturition Date	49
VBIRESTD	9(1)	N	1	Date estimated?	49
VBIRTERM	X(1)	A	1	Pregnancy termination reason	49
VBIRASST	9(1)	N	1	Assistance given during parturition?	49
VBIRCNTA	9(2)	N	2	Parturition count (no. animals born)	50

New Born Animal (3 times)

VBIRBORN	9(2)	N	2	Order born	65
VBIRSEXB	X(1)	A	1	Sex of animal at birth	65
VBIRFATE	X(1)	A	1	Fate of animal at birth	65
VBIRKABI	9(9)	N	9	New born animal key	

Female Reproductive Status

V019RDAT	9(6)	N	6	Reproduction Status Date	43
V019MTHD	9(1)	N	1	Reproduction Status Method	43
V019RSTS	9(1)	N	1	Reproductive Status	43

Female Mating (10 times)

V020MDAT	9(6)	N	6	Mating Date	45
V020MTYP	9(1)	N	1	Mating Type	45
V020ANML	9(9)	N	9	Mate Key (male)	46

Animal Lactation

V026-NUML	9(2)	N	2	Lactation number	55
V026-CTSL	X(1)	A	1	Lactation start reason	55
V026-PDAT	9(7)	N	7	Lactation parturition date	55
V026-FDAT	9(7)	N	7	Lactation start date	55
V026-LDAT	9(7)	N	7	Lactation end date	57
V026-DIML	9(3)	N	3	Lact-dim (production measure)	56
V026-AML3	9(6)V9	N	7	Volume of milk (l) for first 305 days	56
V026-AMLX	9(6)V9	N	7	Volume after 305 days	56
V026-AFL3	9(3)V99	N	5	Milk fat (kg) in 305 days	56
V026-AFLX	9(3)V99	N	5	Milk fat (kg) after 305 days	56
V026-APL3	9(3)V99	N	5	Milk protein (kg) in 305 days	56
V026-APLX	9(3)V99	N	5	Milk protein (kg) after 305 days	56
V026-NUMT	9(3)	N	3	Lactation number of tests	57
V026-NPIL	9(3)	N	3	Lactation latest PI	57
V026-HAVG	9(1)	N	1	Exclude from herd lactation averages	58
V026-IEPI	9(1)	N	1	Exclude from herd PI averages	58
V026-RSNA	X(2)	A	2	Dry off reason	58
V026-NLLP	9(2)	N	2	Lifetime PI Lactation number	58
V026-DIMX	9(3)	N	3	Number of days used in the excess calc	56
V026-NORT	9(3)	N	3	Number of normal tests this lactation	59

WLR = within lifetime reliability

V026-PIFI	9(4)V9	N	5	Initial fat PI	59
V026-PLFI	9(3)	N	3	Initial WLR of fat PI	59
V026-PIPI	9(4)V9	N	5	Initial protein PI	
V026-PLPI	9(3)	N	3	Initial WLR of protein PI	59
V026-PIMI	9(4)V9	N	5	Initial milk volume PI	59
V026-PLMI	9(3)	N	3	Initial WLR for milk volume PI	59
V026-BIFI	9(4)V9	N	5	Initial fat BI	60
V026-BLFI	9(3)	N	3	Initial WLR for fat BI	60

AR = ancestry reliability

V026-BAFI	9(3)	N	3	Initial AR for fat BI	60
V026-BIPI	9(4)V9	N	5	Initial protein BI	60
V026-BLPI	9(3)	N	3	Initial WLR of protein BI	60
V026-BAPI	9(3)	N	3	Initial AR of protein BI	60
V026-BIMI	9(4)V9	N	5	Initial milk volume BI	60
V026-BLMI	9(3)	N	3	Initial WLR of milk volume BI	61
V026-BAMI	9(3)	N	3	Initial AR of milk volume BI	61

V026-PIFT	9(4)V9	N	5	Fat PI	61
V026-PRFT	9(3)	N	3	Fat PI reliability	61
V026-PIPT	9(4)V9	N	5	Protein PI	61
V026-PRPT	9(3)	N	3	Protein PI reliability	61
V026-PIMK	9(4)V9	N	5	Milk volume PI	61

V026-PRMK	9(3)	N	3	Milk volume PI reliability	61
V026-PIPY	9(4)V9	N	5	Payment PI	62
V026-PRPY	9(3)	N	3	Payment PI reliability	62
V026-BIFT	9(4)V9	N	5	Fat BI	62
V026-BRFT	9(3)	N	3	Fat BI reliability	62
V026-BIPT	9(4)V9	N	5	Protein BI	62
V026-BRPT	9(3)	N	3	Protein BI reliability	62
V026-BIMK	9(4)V9	N	5	Milk volume BI	62
V026-BRMK	9(3)	N	3	Milk volume BI reliability	63
V026-BIPY	9(4)V9	N	5	Payment BI	63
V026-BRPY	9(3)	N	3	Payment BI reliability	63
V026-FTIX	9(4)V9	N	5	Fat index	63
V026-PRIX	9(4)V9	N	5	Protein Index	63
V026-MIIX	9(4)V9	N	5	Milk volume index	63

Test Day Production Detail (12 times)

V262-TDAT	9(7)	N	7	Test Date	275
V262-CDAT	9(7)	N	7	Change Date	275
V262-CRNI	9(2)	N	2	Reason no inspection	276
V262-CRNU	9(2)	N	2	Non sample use reason	276
V262-CMAA	9(2)	N	2	Milk abnormal	276
V262-INDA	X(1)	A	1	Assessment indicator	276
V262-AMAA	9(2)V9	N	3	Amount of milk, AM	276
V262-AMAP	9(2)V9	N	3	Amount of milk, PM	277
V262-AMA3	9(2)V9	N	3	Amount of milk, 3rd milking	277
V262-PCFA	9(3)V99	N	5	% milkfat	277
V262-PCPA	9(3)V99	N	5	% protein	277
V262-PCLA	9(3)V99	N	5	% lactose	277
V262-NSCC	9(5)	N	5	Somatic cell count	278
V262-NPEL	9(3)	N	3	Production estimate	278
V262-INDD	X(1)	A	1	Invalid data	278
V262-CGRP	9(2)	N	2	Contemporary group	278
V262-PIFT	9(4)V9	N	5	Fat PI	278
V262-PRFT	9(3)	N	3	Fat PI reliability	278
V262-PIPT	9(4)V9	N	5	Protein PI	279
V262-PRPT	9(3)	N	3	Protein PI reliability	279
V262-PIMK	9(4)V9	N	5	Milk volume PI	279
V262-PRMK	9(3)	N	3	Milk volume PI reliability	279
V262-PIPY	9(4)V9	N	5	Payment PI	279
V262-PRPY	9(3)	N	3	Payment PI reliability	279
V262-BIFT	9(4)V9	N	5	Fat BI	279
V262-BRFT	9(3)	N	3	Fat BI reliability	279
V262-BIPT	9(4)V9	N	5	Protein BI	279
V262-BRPT	9(3)	N	3	Protein BI reliability	280
V262-BIMK	9(4)V9	N	5	Milk volume BI	280
V262-BRMK	9(3)	N	3	Milk volume BI reliability	280
V262-BIPY	9(4)V9	N	5	Payment BI	280
V262-BRPY	9(3)	N	3	Payment BI reliability	280
V262-TRFT	9(3)V9(4)	N	7	Transformed fat value	280
V262-TRPT	9(3)V9(4)	N	7	Transformed protein value	280
V262-TRMK	9(3)V9(4)	N	7	Transformed milk value	280
V262-GRPW	9(1)V9(4)	N	5	Contemporary group weight	281
V262-MPLN	X(4)	A	4	????	
V262-MEGL	X(3)	A	3	????	
V262-MNGL	X(3)	A	3	????	
V262-HERD	9(2)	N	2	Herd Number	275
V262-TSPL	9(2)	N	2	LIA production season	
V262-NTSH	9(3)	N	3	HT service visit number	276

Non production traits survey

VSVYCPTI	X(4)	A	4	Inspector	349

VSVYDINT	9(6)	N	6	Inspection date	350
VSVYVT90	9(1)	N	1	Adaptability to milking	350
VSVYVT91	9(1)	N	1	Shed temperament	350
VSVYVT92	9(1)	N	1	Milking speed	350
VSVYVT93	9(1)	N	1	Farmer opinion	350
VSVYVT94	9(1)	N	1	Weight	351
VSVYVT95	9(1)	N	1	Stature	351
VSVYVT96	9(1)	N	1	Dairy Capacity	351
VSVYVT97	9(1)	N	1	Rump angle	351
VSVYVT98	9(1)	N	1	Rump Width	351
VSVYVT99	9(1)	N	1	Legs	351
VSVYVT100	9(1)	N	1	Udder support	351
VSVYVT101	9(1)	N	1	Fore udder	351
VSVYVT102	9(1)	N	1	Rear udder	352
VSVYVT103	9(1)	N	1	Front teat placement	352
VSVYVT104	9(1)	N	1	Back teat placement	352
VSVYVT105	9(1)	N	1	Udder overall	352
VSVYVT106	9(1)	N	1	Body conformation	352
VSVYCN01	X(2)	A	2	Comment	352
VSVYCN02	X(2)	A	2	Comment	352
VSVYCN03	X(2)	A	2	Comment	352
VSVYCN04	X(2)	A	2	Comment	353
VSVYCN05	X(2)	A	2	Comment	353
VSVYCN06	X(2)	A	2	Comment	353
VSVYCN07	X(2)	A	2	Comment	353
VSVYCN08	X(2)	A	2	Comment	353
VSVYCN09	X(2)	A	2	Comment	353
VSVYCN10	X(2)	A	2	Comment, or classification	353

Animal

Y001BRTH	9(6)	N	6	Animal Date of Birth	2

Animal Cross Breed (3 times)

V002BRED	9(1)	N	1	Breed code	9
V00216TH	9(2)	N	2	Sixteenths of this breed	9

Animal Lactation - Dam

V026BIFT	9(4)V9	N	5	Fat BI	62
V026BRFT	9(3)	N	3	Fat BI reliability	62
V026BIPT	9(4)V9	N	5	Protein BI	62
V026BRPT	9(3)	N	3	Protein BI reliability	62
V026BIMK	9(4)V9	N	5	Milk volume BI	62
V026BRMK	9(3)	N	3	Milk volume BI reliability	63
V026PIFT	9(4)V9	N	5	Fat PI	61
V026PRFT	9(3)	N	3	Fat PI reliability	61
V026PIPT	9(4)V9	N	5	Protein PI	61
V026PRPT	9(3)	N	3	Protein PI reliability	61
V026PIMK	9(4)V9	N	5	Milk volume PI	61
V026PRMK	9(3)	N	3	Milk volume PI reliability	61

Female Parturition - Dam

VBIRPDAT	9(6)	N	6	Parturition Date	49
VBIRESTD	9(1)	N	1	Date estimated?	49
VBIRTERM	X(1)	A	1	Pregnancy termination reason	49
VBIRASST	9(1)	N	1	Assistance given during parturition?	49
VBIRCNTA	9(2)	N	2	Parturition count (no. animals born)	50

New Born Animal - Dam (3 times)

VBIRBORN	9(2)	N	2	Order born	65	
VBIRSEXB	X(1)	A	1	Sex of animal at birth	65	
VBIRFATE	X(1)	A	1	Fate of animal at birth		65
VBIRKABI	9(9)	N	9	New born animal key		

Animal - Dam

Y001SIRE	9(9)	N	9	Animal Sire		1

AB or Grade Bull - Dam

VAIDBULL	9(6)	N	6	????

APPENDIX B

RELATIONAL SCHEMA OF LIC DATABASE

APPENDIX C

LIVESTOCK IMPROVEMENT CORPORATION - INTERVIEW NOTES

12 November 1993

Interviewee: Marine
Livestock Improvement Association

The following information was obtained from Marine:

- Transfer Out Date = date of the fate of the animal.

- The lactation start date and the parturition date should be the same.

Assumption : They can be used in place of the other if one is missing, or one can be eliminated.

- If the milk fat level at testing is zero, then no test took place.

- If the milk fat level is greater than 1000 kgs then the data is likely to be incorrect.

- Within lifetime reliability an estimate of the reliability of the PI and BI readings. This figure is calculated in a complex manner.

- Ancestry reliability means how certain the genetics of the parent is known.

- The payment PI and BI means the payment likely to be received based on the fat content, the protein content of the milk; discounted for the milk volume. Payment is the expected payment over a lifetime based on the three components.

- Production index simplified indicates how much milk the cow will produce in a life time.

- Breeding index how well are the progeny going to be likely to produce milk during their lifetime.

- The fat index, the protein index and the milk volume index are not relevant for year-to-year comparisons since the farmer does not get these.

- Fat, protein and milk volume indexes combine to form the lactation index.

- The lactation index could be ranked within the herd, same for payment PI.

- HT in the data dictionary stands for herd test.

- Milk abnormal V262cmaa is caused by spoilt samples or the cow is sick, so both cows related and testing problem related causes.

- The somatic cell count means if it is high, infection is likely so the lower the better. In particular, the somatic cell count will indicate the amount of mastitis. The guidelines for the levels of somatic cell count can be obtained from the NZ Dairy Group. In 1994 farmers are going to face a high penalty for high somatic cell counts in the milk. We need to partition the data into cell count groupings.

- Transformed value (V262TRFT) is used in the calculation of the PI but is not seen by the farmer.

- Contemporary group is the breed and age classification within the herd. It is used in the calculation of the PI. (V262CGRP) Value of the weight is used in calculation of the PI and is not given.

- VLNMHERD herd number in farm = V262HERD herd number where the test took place.

- LIA Production Season is 1 June to 31 May the following year. e.g. for 1993, the production season is 1/6/93 to 31/5/94.

- If milk volume equals 1.1 litres, then no lactation production is calculated. This situation is because there is not enough normal data to calculate this.

- Year identified came into effect from 1985.

- Production Index is a forecasted figure, which usually should improve with age. There is no adjustment for age in the calculation of the production index.

- Farmers have gone from once considering fat a prime indicator of milk quality to protein over the years. Extra emphasis is placed on payment. Farmers see comment code 10.

- The culling rate per year is around 25%. The reason why 1987/88 data shows only two culled cows, is that culling may be categorised as Dead or Sold under the Fate code. If deaths are greater than 20 per year then the farmer would be a suspicious farmer (or perhaps a natural disaster occurred). The number of fates per year is determined by the current year less the previous years fates. i.e. the number of cows culled in one year will be the number of cows shown culled in the current year less the number of cows culled in the previous year. Therefore, the fate code is accumulative.

- The low numbers of dam and sire details field shows cause for concern. Marine thought this should not be the case.

- For a two year old cow, there will be more data complete than in later years since there is no need to repeat this data in later years. e.g. trait data.

- An adjustment for the month of the test in relation with lactation is taken into account in the indexes.

- Year by year comparisons will only be applicable for the Payment PI and BI.

- Annual income calculation for each cow will also be worth adding as an extra field. The protein payout, the fat payout and the milk volume penalty for each year can be obtained from the Dairy Group.

- Do not worry about PIs and BIs across herd BUT if comparing PIs and BIs across herd, take the age into consideration. The age groupings for cows are 2 years, 3 years, 4 years and mature.

- Consider the rankings within herd of PI's e.g. top & bottom 10.

- The breeding index does change with other related animals being born which give a better indication of the cows breeding.

- The sire is possibly an important factor in the decision. Some farmers keep the offspring of a particular sire.

Livestock Improvement Association are going to provide us with:

- Documentation on the codes for V262CMAA TD MILK ABNM code descriptions (6 used), and V262CRNU TD NON USE SMPL code descriptions.

- Somatic cell count information.

- Contemporary group - information on the exact meaning.

QUESTION:

Why is a high milk volume bad?

January 1994

WXV data:

Q1. Some of the FATE CODES starting in 1989 data files, have a TRANSFER OUT DATE of 1993

A1. Cows in the 1989 file are only those which were current in 1989.

comment: The reply to this question did not make sense later since `late's and `early's are discovered in our status code.

Q2. Is the `G' FATE CODE used as an `other' category for the fate of an animal? Is it possible that this code is used when the fate of an animal is undetermined?

A2. Yes

Q3. Can the PAYMENT PI be multiplied by the milk price (from the NZDG) to give the value of the cows milk in dollars? Or do you have any suggestions as to how to encorporate the milk price into the data?

A3. No, you need to multiply the milk price by the kg of fat, protein, and milk.

Q4. Could we please have descriptions for the following items:

	Answer
Cause of Fate = MT	Not Pregnant
VAID-BULL	Identification for bull AV or grade bull code
VAID-NOAY	Made from previous two attributes to get current Animal_Key of cow i.e. partipant code + year identified
V262-MPLN	Makes up the herd location
V262-MEGL	"
V262-MNGL	"

APPENDIX D

DIARY NOTE OF LECTURER VISIT

Discussion with Massey University Lecurer - Specialist in Clustering

Data:

Do not eliminate any of the fields at this stage. Let ID3 indentify which fields are irrelevant.

Need to ensure that all the relevant data to the decision is present.

Need to create explicit attributes from the data. e.g. differences in fat content for each year. Remove data from being sequential since farmers may use comparisons to determine when to cull. Use ratios, averages, sums, weighted averages to try to remove time relevance. The machine learning tools are not likely to interpret relationships in identifying the importance of attributes. Also look at BI and PI relative to the rest of the herd. Hence important to know exactly what each attribute means.

Data format:

Have only one line of attributes per record (animal). Therefore if there are three years of figures then put the value for each year along the same line.

e.g. Cow1 Lactation_Start_Date_Year1 Lactation_Start-Date_Year2

note This was not a good idea since based on the yearly decision of culling, the lactation start date for the following year will have absolutely no bearing on a decision to cull the cow.

Clustering with Attributes - turn dates into time period measures e.g. no. of years from 1980.

Ensure that null values are not mis-represented as zeros in the field.

Machine Learning Tools:

ID3 will not pick up highly correlated attributes (i.e. those that have similar impact on the decision). To overcome this deficiency, record the impact of an attribute, then re-run the search with that attribute eliminated so it will pick up any other attributes with equal impact on the decision.

As well as using known ML tools, consider using tools which use statistical analysis techniques e.g. regression, correlation.

Other references:

Causal Modelling	Jud Pearl

Recommended Steps:

1. Get rest of years to start putting together large file with comparisons.

2. Check that null values are accurately presented (see Data Format).

3. Run on well-populated records initially to gauge which attributes are important. i.e. select sub-sets of attributes which do not have too many missing values.

4. Need to create data independent attributes. (see Data section.

APPENDIX E

RAW FORMAT TO SPREADSHEET IMPORT FORMAT CONVERSION

rawtoss converts files from LIC database format to a general spreadsheet import format, with comma seperation and text delimited with double quotes.

rawtolotus converts files from raw LIC database format to a comma seperated format suitable for import by lotus1-2-3, with not more than 511 characters per line.

FILES NEEDED FOR USE

In order to use rawtoss and rawtolotus, the following files are needed;

reformat	executable
Format	data file
join.awk	awk script
delimit	executable
split2	executable

INPUT

rawtolotus and rawtoss expect a file with the filename <name>.xtr, and optionally, also a file called <name>.262.

To use rawtolotus, enter

```
rawtolotus <name>
```

To use rawtoss, enter

```
rawtoss <name>
```

at the shell prompt. The extract file will be converted first, and if a v262 file is present, then the data from this file will be joined into the extract.

OUTPUT

RAWTOSS

rawtoss produces a file called <name>x.dat.

eg
To convert the raw files wxv87.xtr and wxv87.262 into a spreadsheet import format file, type "rawtoss wxv87"

rawtoss will produce a file called wxv87x.dat.

This will be comma seperated with double quotes around all text values, and "?" in place of missing values.

RAWTOLOTUS

rawtolotus produces 8 files, named

<filename>0.dat, <filename>1.dat, ..., <filename>7.dat

These files contain 100 attributes each (except <filename>7.dat, which will contain only 5), and the values of all the cases for those attributes.

These files are comma seperated, with double quotes around all text values.

They also have "?" substituted for any missing values in the data.

eg

To convert the raw files wxv87.xtr and wxv87.262 into a lotus import format file, type "rawtolotus wxv87".

rawtolotus will produce 8 files called wxv87x0.dat, wxv87x1.dat, wxv87x2.dat ... wxv87x7.dat.

APPENDIX F

DISCUSSION OF PC SOFTWARE TOOLS USED IN DATA MANIPULATION

The three main requirements of the PC based tools were:
1. The ease of selecting attributes to be included in a test data sub-set.
2. The ease of changing the data e.g. converting existing attributes into linear form.
3. The ability to view the data to improve understanding and test the output of the workbench e.g. query facilities.

Microsoft Access v 1.1 - relational database software

Initially Microsoft Access version 1.1 was used for data munging because of the implications of the data having coming from a relational database. The data was imported from the raw dataset by setting up a database specification. The database specification could be used on all the datasets to ensure that the same structure could be maintained assisting in reducing the chance of error. Once the data was imported, one Access database was created for each farm and each database contained a table for each year. The Access query facility was used to test the results of the workbench, to create sub-sets of data from the the orginal dataset, and to create additional columns based on the raw data in a format for the workbench. Data was exported to an ASCII file with the first record containing the attributes names. The ASCII file was then transferred to the UNIX system where a program converted the file into the special workbench format.

Lotus 123 Spreadsheet

Lotus 123 was used in place of Microsoft Access for munging the data. The importing of the raw data required a conversion process using a UNIX based programme. The first step in converting the raw data involved including a comma delimiter between attribute contents. The second step in conversion involved splitting the file into lots of 100 attributes since Lotus 123 had a 512 character import limit per row. The result was 8 x 100 attribute files by six years for 10 farms - a total of ??? files to be imported. A macro was written in Lotus 123 to import the attribute names from a separate spreadsheet and the eight data files into the Lotus spreadsheets. Each spreadsheet file contained six years and two hundred attributes per farm. So there were 4 spreadsheet files per farm (i.e. 705 attributes/200=4 spreadsheets, refer Spreadsheet Layout). Each year was placed into one sheet (FOOTNOTE: A sheet is the name Lotus 123 uses for one of the 256 possible column by row spreadsheets in its three-dimensional structure). The following limitations determined the way in which the data was imported into each spreadsheet:

5 megabytes of primary memory
256 maximum columns per spreadsheet

The initial attributes were extracted from the main spreadsheets while still retaining the six year three dimensional format. The new attributes which required inter-year comparisions were created first using the @VLOOKUP (refer Discussion of methods) function. To avoid slow operation of a spreadsheet heavily laiden with formulas, an extract of the first data row containing the formulas and an extract of the values of the rest of the rows were performed. The extracts were then split into six annual two-dimensional worksheets to create the additional columns (refer New Attributes List, Appendix ???). Please note, that with a more powerful micro computer alot these extractions would probably be unnecessary. Once the new attributes were created, the spreadsheet was exported to an ASCII file by way of creating a print file with no formatting. The data was then transfered to the UNIX system where a conversion program took the print file and converted into the format for the workbench.

Limitations of the Microsoft Access v 1.1

To setup a database specification required knowledge as to the exact form of the data including the a name for the attribute, the order of the attribute, the type of attribute and

the column number it started at in the ASCII dataset. Some time after switching to Lotus 123 we discovered that the information we had regarding the structure of the dataset was at fault, and some of the figures attributes we put through the workbench from MS Access were erroneous.
Although Microsoft Access was strong on the query facility, in the early version its statisticaly functions were limited and in one instance a bug meant one feature was disfunctional.
One other bug was the fact that Access could not import dates in YYMMDD format. A special conversion effort from text to date format was a time consuming handicap for 60 x 200 record tables.
Another problem with Microsoft Access was the inabiltity of access to accept and operate on question marks in place of missing values. This limitation was especially important in situations where a missing value represented something meaningful.
MS Access provided macros which could be used to create forms (forms were a feature we did not use) but did not appear to have macros which could perform repetitive data manipulation of a similarly structured database. Macros to avoid repetition thus reducing error are common to most other PC based software so alternatives were considered.
The most convenient method of handling the files in Access were placing each year of data into a separate tables. Unfortunately, Microsoft Access is not designed to efficiently handle the 139 attributes of data we initially imported and together with the hardware limitations, using Access became time consuming to perform a query. It was not possible to use the Windows Swap drive since Access could not manage the use of vertual memory without intermittently crashing to the DOS prompt irrespective of what you happened to be doing at the time.
One thing we did identify using Access was the requirement to perform queries and view that data was essential in speeding up the process of understanding the data.
One advantage of MS Access was its ability to export text files with the first record containing the attribute name. This created a flexible method of exporting data in preparation for the workbench.

The Lotus 123 Advantage

The use of the three dimensional feature of Lotus 123 meant time was saved by performing operations on all six years for each farm in one action. With a more powerful microcomputer it may have been possible to operate on all 10 farms and the two hundred attributes for each farm in one action. With the three dimensional structure it was possible to add/delete attributes across years in one action and copy formulas down columns sheet A to sheet F in six actions.
One other feature proved invaluable was the @VLOOKUP function. This allowed us to cope work around the problem of ANIMAL KEYS not matching across years. This problem was understandbly the cows entering or leaving the farm during the six year period. Using a query (Access method) we could only do inter-year comparisons easily if the cows were common to all six years, but using the @VLOOKUP function this formula would take the Animal_Key in the current year (sheet), look for it in a range in a previous year (sheet), and select the contents of any column from the cows record in the previous year. If there was no matching Animal_Key in the previous year, a `?' was placed in the column. Thus we were able to get the previous years production and breeding index contained in the same record as the current year for comparison.
Lotus 123 has a database query facility which is slightly more complex to use than Access, but was more functional and flexible than the Access query facility.
Another advantage of Lotus 123 was the ease and speed of extracting and combine data across specified ranges across spreadsheet files. In one action any number of attributes could be extracted in three dimension form (i.e. all six years) into the current spreadsheet from a six megabyte spreadsheet file, within reasonable time.
The @IF function enabled us to derive workbench format attributes from other attributes. The most critical of these attributes being our STATUS_CODE used to determine whether the cow was retained on the farm or whether it was removed on the farm a the time the data about the cow was recorded.
One limitation of Lotus 123 was the inability to export to ASCII files in a comma delimitered format.

The ability to view the output in front of your eyes in two-dimensional form immediately. This advantage is expecially useful in detection of errors when creating new attributes or changing the data in anyway.
A facility to audit the process you have used to obtain the results by viewing formulas used in the process and then being able to make minor adjustments was an advantage using Lotus 123.
If the workbench is eventually to be targetted at business, the existance of a copy of Lotus 123 is highly probably, or will quickly be obtained. A budget of $1000 for a copy of Lotus is a cheap alternative to someone experienced in UNIX and Awk.
Lotus 123 comes complete with statistical functions and graph functions which compete with even most statistical command-based spreadsheets.
The evolution of data communications is improving the communication between UNIX and PC based platforms e.g. Unixware. So PC based 123 and the workbench are not seen as rivals but compliments.
The incorporation of the three dimensional spreadsheet has meant that alike datasets could be operated on in one action as apposed to running one program on each individual file.
One disadvantage of Lotus 123 over using a programming language is Lotus is slow at performing manipulation of large datasets given the equivalent hardware contraints.
The second disadvantage of Lotus 123 is a column limitation of 256 attributes.
The third limitation is the maximum number of records a spreadsheet file can hold is subject to hardware memory contraints.

APPENDIX G

DATA DICTIONARY OF WEKA (NEW) ATTRIBUTES

WEKA_Fat_PI

derived from : Fat_PI

The FAT PI with zero values replaced by `?', since a FAT PI of zero indicates an unsuccessful test or a dead cow, not an actual PI measure of zero.

WEKA_Protein_PI

derived from: Protein_PI

The PROTEIN PI with zero values replaced by `?', since a PROTEIN PI of zero indicates an unsuccessful test or a dead cow, not an actual PI measure of zero.

WEKA_MilkVolume_PI

derived from: MilkVolume_PI

The MILK VOLUME PI with zero values replaced by `?', since a MILK VOLUME PI of zero indicates an unsuccessful test or a dead cow, not an actual PI measure of zero.

WEKA_Payment_PI

derived from: Payment_PI

The PAYMENT PI with zero values replaced by `?', since a PAYMENT PI of zero indicates an unsuccessful test or a dead cow, not an actual PI measure of zero.

WEKA_Fat_BI

derived from: Fat_BI

The FAT BI with zero values replaced by `?', since a FAT BI of zero indicates an unsuccessful test or a dead cow, not an actual BI measure of zero.

WEKA_Protein_BI

derived from: Protein_BI

The PROTEIN BI with zero values replaced by `?', since a PROTEIN BI of zero indicates an unsuccessful test or a dead cow, not an actual BI measure of zero.

WEKA_MilkVolume_BI

derived from: MilkVolume_BI

The MILK VOLUME BI with zero values replaced by `?', since a MILK VOLUME BI of zero indicates an unsuccessful test or a dead cow, not an actual BI measure of zero.

WEKA_Payment_BI

derived from: Payment_BI

The PAYMENT BI with zero values replaced by `?', since a PAYMENT BI of zero indicates an unsuccessful test or a dead cow, not an actual BI measure of zero.

WEKA_Prev_FatPI

derived from: WEKA_Fat_PI

The WEKA FAT PI from the previous year's data for the same cow i.e. the WEKA PREV FAT PI in 1998 for cow A is the value in the WEKA FAT PI in 1987 for cow A. The 1987 WEKA PREV FAT PI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_Prev_ProteinPI

derived from: WEKA_Protein_PI

The WEKA PROTEIN PI from the previous year's data for the same cow i.e. the WEKA PREV PROTEIN PI in 1998 for cow A is the value in the WEKA PROTEIN PI in 1987 for cow A. The 1987 WEKA PREV PROTEIN PI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_Prev_MilkVolume_PI

derived from: WEKA_MilkVolume_PI

The WEKA MILK VOLUME PI from the previous year's data for the same cow i.e. the WEKA MILK VOLUME FAT PI in 1998 for cow A is the value in the WEKA MILK VOLUME PI in 1987 for cow A. The 1987 WEKA PREV MILK VOLUME PI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_Prev_PaymentPI

derived from: WEKA_Payment_PI

The WEKA PAYMENT PI from the previous year's data for the same cow i.e. the WEKA PAYMENT FAT PI in 1998 for cow A is the value in the WEKA PAYMENT PI in 1987 for cow A. The 1987 WEKA PREV PAYMENT PI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_Prev_FatBI

derived from: WEKA_Fat_BI

The WEKA FAT BI from the previous year's data for the same cow i.e. the WEKA PREV FAT BI in 1998 for cow A is the value in the WEKA FAT BI in 1987 for cow A. The 1987 WEKA PREV FAT BI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_Prev_ProteinBI

derived from: WEKA_Protein_BI

The WEKA PROTEIN BI from the previous year's data for the same cow i.e. the WEKA PREV PROTEIN BI in 1998 for cow A is the value in the WEKA PROTEIN BI in 1987 for cow A. The 1987 WEKA PREV PROTEIN BI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_Prev_MilkVolume_BI

derived from: WEKA_MilkVolume_BI

The WEKA MILK VOLUME BI from the previous year's data for the same cow i.e. the WEKA MILK VOLUME FAT BI in 1998 for cow A is the value in the WEKA MILK VOLUME BI in 1987 for cow A. The 1987 WEKA PREV MILK VOLUME BI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_Prev_PaymentBI

derived from: WEKA_Payment_BI

The WEKA PAYMENT BI from the previous year's data for the same cow i.e. the WEKA PAYMENT FAT BI in 1998 for cow A is the value in the WEKA PAYMENT BI in 1987 for cow A. The 1987 WEKA PREV PAYMENT BI contains a `?' since there was no data for 1986. The @VLOOKUP command in Lotus 123 was used to derive this attribute.

WEKA_FatPI_Change

derived from: WEKA_FatPI
WEKA_Prev_FatPI

The change in the FAT PI in a over one year time period. A positive number indicates an increase in the production index (PI), while a negative number indicates a decrease in the production index. A `?' indicates a missing value in either the current year production index or the previous year production index.

If the WEKA FAT PI or the WEKA PREV FAT PI contains a `?' then the WEKA FATPI CHANGE is a `?'. If the WEKA FAT PI and the WEKA PREV FAT PI both contain a values, then the WEKA PREV FAT PI is subtracted from the WEKA FAT PI.

WEKA_ProteinPI_Change

derived from: WEKA_ProteinPI
WEKA_Prev_ProteinPI

The change in the PROTEIN PI in a over one year time period. A positive number indicates an increase in the production index (PI), while a negative number indicates a decrease in the production index. A `?' indicates a missing value in either the current year production index or the previous year production index.

If the WEKA PROTEIN PI or the WEKA PREV PROTEIN PI contains a `?' then the WEKA PROTEINPI CHANGE is a `?'. If the WEKA PROTEIN PI and the WEKA PREV PROTEIN PI both contain a values, then the WEKA PREV PROTEIN PI is subtracted from the WEKA PROTEIN PI.

WEKA_Milk VolumePI_Change

derived from: WEKA_MilkVolumePI
WEKA_Prev_MilkVolumePI

The change in the MILK VOLUME PI in a over one year time period. A positive number indicates an increase in the production index (PI), while a negative number indicates a decrease in the production index. A `?' indicates a missing value in either the current year production index or the previous year production index.

If the WEKA MILK VOLUME PI or the WEKA PREV MILK VOLUME PI contains a `?' then the WEKA MILK VOLUME PI CHANGE is a `?'. If the WEKA MILK VOLUME PI and the WEKA PREV MILK VOLUME PI both contain a values, then the WEKA PREV MILK VOLUME PI is subtracted from the WEKA MILK VOLUME PI.

WEKA_PaymentPI_Change

derived from: WEKA_PaymentPI
WEKA_Prev_PaymentPI

The change in the PAYMENT PI in a over one year time period. A positive number indicates an increase in the production index (PI), while a negative number indicates a decrease in the production index. A `?' indicates a missing value in either the current year production index or the previous year production index.

If the WEKA PAYMENT PI or the WEKA PREV PAYMENT PI contains a `?' then the WEKA PAYMENT PI CHANGE is a `?'. If the WEKA PAYMENT PI and the WEKA PREV PAYMENT PI both contain a values, then the WEKA PREV PAYMENT PI is subtracted from the WEKA PAYMENT PI.

WEKA_FatBI_Change

derived from: WEKA_FatBI
WEKA_Prev_FatBI

The change in the FAT BI in a over one year time period. A positive number indicates an increase in the breeding index (BI), while a negative number indicates a decrease in the breeding index. A `?' indicates a missing value in either the current year breeding index or the previous year breeding index.

If the WEKA FAT BI or the WEKA PREV FAT BI contains a `?' then the WEKA FATBI CHANGE is a `?'. If the WEKA FAT BI and the WEKA PREV FAT BI both contain a values, then the WEKA PREV FAT BI is subtracted from the WEKA FAT BI.

WEKA_ProteinBI_Change

derived from: WEKA_ProteinBI
WEKA_Prev_ProteinBI

The change in the PROTEIN BI in a over one year time period. A positive number indicates an increase in the breeding index (BI), while a negative number indicates a decrease in the breeding index. A `?' indicates a missing value in either the current year breeding index or the previous year breeding index.

If the WEKA PROTEIN BI or the WEKA PREV PROTEIN BI contains a `?' then the WEKA PROTEINBI CHANGE is a `?'. If the WEKA PROTEIN BI and the WEKA PREV PROTEIN BI both contain a values, then the WEKA PREV PROTEIN BI is subtracted from the WEKA PROTEIN BI.

WEKA_MilkVolumeBI_Change

derived from: WEKA_MilkVolumeBI
WEKA_Prev_MilkVolumeBI

The change in the MILK VOLUME BI in a over one year time period. A positive number indicates an increase in the breeding index (BI), while a negative number indicates a decrease in the breeding index. A `?' indicates a missing value in either the current year breeding index or the previous year breeding index.

If the WEKA MILK VOLUME BI or the WEKA PREV MILK VOLUME BI contains a `?' then the WEKA MILK VOLUME BI CHANGE is a `?'. If the WEKA MILK VOLUME BI and the WEKA PREV MILK VOLUME BI both contain a values, then the WEKA PREV MILK VOLUME BI is subtracted from the WEKA MILK VOLUME BI.

WEKA_PaymentBI_Change

derived from: WEKA_PaymentBI
WEKA_Prev_PaymentBI

The change in the PAYMENT BI in a over one year time period. A positive number indicates an increase in the breeding index (BI), while a negative number indicates a decrease in the breeding index. A `?' indicates a missing value in either the current year breeding index or the previous year breeding index.

If the WEKA PAYMENT BI or the WEKA PREV PAYMENT BI contains a `?' then the WEKA PAYMENT BI CHANGE is a `?'. If the WEKA PAYMENT BI and the WEKA PREV PAYMENT BI both contain a values, then the WEKA PREV PAYMENT BI is subtracted from the WEKA PAYMENT BI.

WEKA_Year

Each record contains some data about the animal relating to one specific year of its life e.g. the production index (PI) for the cow in 1987. The WEKA YEAR is the year the data was collected about each animal.

WEKA_Age

derived from : Animal_Date_of_Birth

The age of the animal at the end of the data year provided there is an Animal_Date_of_Birth recorded.

If the Animal_Date_of_Birth is greater than zero, then 10,000 is added to the data year (in the form YY0000), subtracts Animal_Date_of_Birth, divides by 10,000 and rounds the result to zero decimal places.

If the Animal_Date_of_Birth is equal to zero then a ? is shown as the WEKA_AGE.

WEKA_Calved

derived from : Reproductive_Status

Whether or not the cow has given birth to a calf during the data year or not.

If the Reproductive_Status code is `1' or `3', then the animal has not given birth to a calf, otherwise the animal has given birth to a calf.

Y	The animal has given birth to a calf
N	The animal has not given birth to a calf

WEKA_FatPI_AvgDiff

derived from : WEKA_FatPI

The difference between the WEKA_FatPI and the average fat PI for the participant for the data year.

The average fat PI is calculated by adding the total PI values for the participant for one year, and dividing by the number of PI values greater than zero.

positive number	above average PI
negative number	below average PI

If the WEKA_Fat_PI is missing, then a "?" is shown as the average difference.

WEKA_ProteinPI_AvgDiff

derived from : WEKA_ProteinPI

The difference between the WEKA_ProteinPI and the average Protein PI for the participant for the data year.

The average Protein PI is calculated by adding the total PI values for the participant for one year, and dividing by the number of PI values greater than zero.

positive number	above average PI
negative number	below average PI

If the WEKA_Protein_PI is missing, then a "?" is shown as the average difference.

WEKA_MilkVolumePI_AvgDiff

derived from : WEKA_MilkVolumePI

The difference between the WEKA_MilkVolumePI and the average MilkVolume PI for the participant for the data year.

The average MilkVolume PI is calculated by adding the total PI values for the participant for one year, and dividing by the number of PI values greater than zero.

positive number	above average PI
negative number	below average PI

If the WEKA_MilkVolume_PI is missing, then a "?" is shown as the average difference.

WEKA_PaymentPI_AvgDiff

derived from : WEKA_PaymentPI

The difference between the WEKA_PaymentPI and the average Payment PI for the participant for the data year.

The average Payment PI is calculated by adding the total PI values for the participant for one year, and dividing by the number of PI values greater than zero.

positive number	above average PI
negative number	below average PI

If the WEKA_Payment_PI is missing, then a "?" is shown as the average difference.

WEKA_FatBI_AvgDiff

derived from : WEKA_FatBI

The difference between the WEKA_FatBI and the average fat BI for the participant for the data year.

The average fat BI is calculated by adding the total BI values for the participant for one year, and dividing by the number of BI values greater than zero.

positive number	above average BI
negative number	below average BI

If the WEKA_Fat_BI is missing, then a "?" is shown as the average difference.

WEKA_ProteinBI_AvgDiff

derived from : WEKA_ProteinBI

The difference between the WEKA_ProteinBI and the average Protein BI for the participant for the data year.

The average Protein BI is calculated by adding the total BI values for the participant for one year, and dividing by the number of BI values greater than zero.

positive number	above average BI
negative number	below average BI

If the WEKA_Protein_BI is missing, then a "?" is shown as the average difference.

WEKA_MilkVolumeBI_AvgDiff

derived from : WEKA_MilkVolumeBI

The difference between the WEKA_MilkVolumeBI and the average MilkVolume BI for the participant for the data year.

The average MilkVolume BI is calculated by adding the total BI values for the participant for one year, and dividing by the number of BI values greater than zero.

positive number	above average BI
negative number	below average BI

If the WEKA_MilkVolume_BI is missing, then a "?" is shown as the average difference.

WEKA_PaymentBI_AvgDiff

derived from : WEKA_PaymentBI

The difference between the WEKA_PaymentBI and the average Payment BI for the participant for the data year.

The average Payment BI is calculated by adding the total BI values for the participant for one year, and dividing by the number of BI values greater than zero.

positive number	above average BI
negative number	below average BI

If the WEKA_Payment_BI is missing, then a "?" is shown as the average difference.

WEKA_TransferOut_Date

derived from :	Transfer_In_Date(1-3)
	Transfer_Out_Date(1-3)

The date the animal transferred out of the herd permanently (i.e. not just a transfer out to another herd)

0 (zero)	retained

If the Transfer_In_Date1 is zero, then `?' (since the first transfer in date usually indicates when the animal first appeared on the farm so no transfer out date is likely to be an error),

or else if the the Transfer_Out_Date1 is zero, then 0 (assumption: if there is no transfer out date following a transfer in date, then the animal is retained),

or else if the Transfer_In_Date2 is zero, then take the Transfer_Out_Date1 to be the WEKA_Transfer_Out_Date (assumption: if there is not transfer in date following a transfer out date, then the previous transfer out date was when the animal was permanently removed from the herd),

or else if Transfer_Out_Date2 is zero, then zero (assumption: if there is no transfer out date following a transfer in date, then the animal is retained),

or else if Transfer_In_Date3 is zero, then Transfer_Out_Date2 (assumption: if there is not transfer in date following a transfer out date, then the previous transfer out date was when the animal was permanently removed from the herd),

or else if Transfer_Out_Date3 is zero, then zero (assumption: if there is no transfer out date following a transfer in date, then the animal is retained),

or else Transfer_Out_Date3.

WEKA_Cow_Retained/Departed

Determines the location of the cow during the data year.

retained - retained on the farm (with the participant) during the data year

departed - removed permanently from the herd during the data year

early - the data year is before the cow transferred into the farm

late - the animal had permanently left the farm before the data year

? - no transfer in date1

If the Transfer_In_Date1 is zero, then `?'

or else if the Transfer_In_Date1 is greater than the data year, then `early',

or else if the WEKA_TransferOut_Date is zero then the cow is `retained',

or else if the WEKA_TransferOut_Date is less than the data year then `late',

or else if the WEKA_TransferOut_Date is after the data year, the animal is `retained',

or else if the WEKA_TransferOut_Date is during the data year, then the cow has `departed' from the herd in the data year.

WEKA_Cause_of_Fate

derived from :	Cause_of_Fate(1-3)
	WEKA_Cow_Retained/Departed

The last of the three Cause_of_Fate codes given that the animal departed permanently from the farm during the data year.

none	animal is retained

If the WEKA_Cow_retained/departed is `retained', then `none'

or else if Cause_of_Fate3 does not contain a missing value then Cause_of_Fate3,

or else if Cause_of_Fate2 does not contain a missing value, then Cause_of_Fate2,

or else Cause_of_Fate1

WEKA_Bad_Event

derived from : WEKA_Cause_of_Fate

Determines whether or not the departure is caused by a bad event which struck the animal (act of God), whether the departure was caused by a decision made by the farmer, or whether the animal was retained.

N - Retained or decision made by the farmer to remove animal.
(WEKA_Cause_of_Fate = OC OA LP SK TE etc)

Y - A random bad event has occurred causing the cow to depart from the herd.
(WEKA_Cause_of_Fate = MF MT EC MA UD BL FT IN CT)

This formula was edited for each herd as new Cause of Fate codes appeared.

If WEKA_Cause_of_Fate is `?', then `?'

or else if the WEKA_Cause_of_Fate is `none' then `N'

or else if the WEKA_Cause_of_Fate is a caused by a bad event (Act of God) then `Y' or else `N'.

WEKA_Status_Code (The Classification)

derived from : WEKA_Cow_Departed/Retained
WEKA_Bad_Event

This code determines which animals were retained, departed, early or late during the data year; and for those animals which were departed, whether they were removed because of a random event (Act of God) or removed because the farmer made an active decision.

dep_random - departure caused by random event (Act of God)

dep_farmer - departure caused by the farmer's decision retained retained on the farm (with the participant) during the data year

early - the data year is before the cow transferred into the farm

late - the animal had permanently left the farm before the data year

If the WEKA_Cow_Departed/Retained is retained, then `retained'

or else if WEKA_Cow_Departed/Retained is `late', then `late'

or else if WEKA_Cow_Departed/Retained is `early', then `early'

or else if WEKA_Cow_Departed/Retained is `departed' and WEKA_Bad_Event is `Y', then dep_random or else if WEKA_Cow_Departed/Retained is `departed' and WEKA_Bad_Event is `N', then dep_farmer.

APPENDIX H

SPREADSHEET EXPORT FORMAT TO ARFF FORMAT FILE CONVERSION

Two means are provided for converting from a spreadsheet export format to ARFF format, flotus and tlotus.

FILES REQUIRED FOR USE

To use tlotus, flotus and qlotus, the following files are needed;

tlotus	shell script
flotus	shell script
qlotus	shell script
attachtitle	shell script
createarff	shell script
slapem	executable
analyze	executable
titles2.awk	awk script

EXPECTED FILENAMES

In order for flotus and tlotus to work properly, they expect filenames to be in a specific format. Both flotus and tlotus expect datafile names to be in the form;

<name><digit>.123 eg, wxv0.123, wxv1.123, etc

Additionally, tlotus expects titles files to be in the form

<name><digit>title.123 eg, wxv0title.123, wxv1title.123

FLOTUS

flotus is used when the attribute names are included in the datafile as the first line of data. flotus expects a whitespace and/or comma seperated file, with cases (including the first line containing the attribute names)
terminated with a newline character. Quotation marks (double or single) are ignored, as they are treated as text delimiters.

If there are multiple datafiles, flotus will abut them into one big ARFF file,
called <name>.arff

eg

To create an ARFF file from the datafiles wxv0.123, wxv1.123, wxv2.123, type "flotus wxv".

flotus will abut the three files and produce an ARFF file called
wxv.arff.

TLOTUS

tlotus is used when the attribute names are in seperate files, each on a seperate line. As with flotus, tlotus expects datafiles to have the filename;

<name><digit>.123

For each datafile, tlotus also expects a corresponding title file, containing the attribute names for that datafile.

tlotus produces a file called <name>a.arff eg

To create an ARFF file from the files
wxv0.123, wxv0title.123, wxv1.123, wxv1title.123, type "tlotus wxv"

tlotus will create an ARFF file called wxva.arff

QLOTUS

qlotus is used when there are multiple datafiles, but the attribute names are all in a single file, one on each line. qlotus will expect a file called

<name>title.123

and produces a file called <name>.arff

eg

To create an ARFF file from the files wxvtitle.123, wxv0.123, wxv1.123, wxv2.123, type "qlotus wxv"

qlotus will create an ARFF file called wxv.arff

APPENDIX I

(paper)