An Entropy Gain Measure of Numeric Prediction Performance

Leonard Trigg

February 4, 1998

1 Current Evaluation Measures

Categorical classifier performance is typically evaluated with respect to error rate, expressed as a percentage of test instances that were not correctly classified. When a classifier produces multiple classifications for a test instance, the prediction is counted as incorrect (even if the correct class was one of the predictions). Although commonly used in the literature, error rate is a coarse measure of classifier performance, as it is based only on a single prediction offered for a test instance. Since many classifiers can produce a class distribution as a prediction, we should use this to provide a better measure of how much information the classifier is extracting from the domain.

Numeric classifiers are a relatively new development in machine learning, and as such there is no single performance measure that has become standard. Typically these machine learning schemes predict a single real number for each test instance, and the error between the predicted and actual value is used to calculate a myriad of performance measures such as correlation coefficient, root mean squared error, mean absolute error, relative absolute error, and root relative squared error. With so many performance measures it is difficult to establish an overall performance evaluation.

The next section describes a performance measure for machine learning schemes that attempts to overcome the problems with current measures. In addition, the same evaluation measure is used for categorical and numeric classifiers.

2 Entropy Gain Evaluation Measure

The entropy gain measure is effectively a measure of the savings (in bits) when the test instance class value (categorical or numeric) is encoded using the scheme's predictions as opposed to encoding the value using a naive method with no knowledge of the training data. This requires that the scheme produces a probability distribution over the possible class values. Unfortunately, most schemes do not output a predicted distribution over class values, instead making a prediction as to the most likely class value(s). There is no simple way to transform these predictions into a probability distribution, for two reasons. First, there is no indication of the relative likelihood of the predicted classes. Second, there is no indication of the likelihood of the classes not predicted.

Cleary *et al.* (1996) describes an adaptive solution to the second problem, and applies the entropy gain measure to machine learning schemes that produce rules. This section describes a modification of this measure for any machine learning scheme that infers a probability distribution over categorical classes for each instance, as well as a comparable measure for machine learning schemes that predict numeric classes.

The number of bits required to encode the category of an instance of class c with respect to a probability distribution P is $-log_2(P(c))$. One naive method for obtaining a class probability distribution is simply to count the number of instances of each class that appear in the observed data. However, if a class does not appear in the observed data, it is assigned a probability of 0, which requires an infinite number of bits to encode. This is called the *zero frequency problem*, discussed further in Witten and Bell (1991). To circumvent this, the counts for each possible class start from 1. If there are N observed instances and n possible classes (each of which occurs f_c times in the observed data), the probability assigned to class c is

$$P_{naive}(c) = \frac{f_c + 1}{N + n}.$$

Note that this probability is independent of the current test instance. The associated entropy is

$$entropy_{naive}(c) = -log_2(P_{naive}(c)).$$

The zero frequency problem can also occur with the probability distribution provided by the classifier. To counter this, the scheme's probability distribution is combined with the naive distribution so that the relative probabilities assigned by the scheme are preserved. Two weights α and β control the blending. α and β are initially assigned probabilities of $\frac{1}{n}$. After each prediction, α is incremented by $P_{scheme}(c|a)$, and β is incremented by $P_{naive}(c)$. If the scheme is consistently providing more accurate predictions than the naive method, $\alpha \gg \beta$, and so α will dominate the weighting below. The modified predicted probability for the class of instance a is calculated as

$$P'(c|a) = \frac{\alpha P_{scheme}(c|a) + \beta P_{naive}(c)}{\alpha + \beta},$$

where $P_{scheme}(c|a)$ is the probability distribution predicted by the classifer. The corresponding entropy is

$$entropy_{scheme}(c|a) = -log_2(P'(c|a)).$$

The entropy gain measure is defined as the difference between $entropy_{naive}$ and $entropy_{scheme}$, averaged over all test instances.

$$\begin{aligned} EG_{scheme} &= \sum_{i=1}^{N} \frac{entropy_{naive}(c_i|a_i) - entropy_{scheme}(c_i|a_i)}{N} \\ &= \sum_{i=1}^{N} \frac{-log_2(P_{naive}(c|a)) + log_2(P_{scheme}(c|a))}{N}. \end{aligned}$$

Although an absolute difference between $entropy_{naive}$ and $entropy_{scheme}$ intuitively makes more sense, we use the average in order to permit comparison over varying numbers of test instances. A negative entropy gain implies the classifier performed no better than predicting the class based solely on the observed class distribution as testing progresses. A positive entropy gain implies the classifier has successfully captured domain information when making its predictions. It is possible for a scheme to achieve a high error rate in a domain, and at the same time perform well with regards to the entropy gain. For example, when a classifier provides multiple classifications for an instance, these are typically regarded as incorrect when calculating error rates. Allocating equal probability to multiple classes can improve the entropy gain measure if the scheme has managed to effectively determine which classes are unlikely. Similarly, if a scheme would have produced the actual class as a "second choice", the error rate would increase, but the entropy gain measure would not necessarily decrease, particularly if the scheme has successfully determined which classes are unlikely. For some problems there will be a simple correspondence between these two measures—if one scheme has a lower error rate than another scheme, it will typically also have a higher entropy gain (Cleary *et al.*, 1996).

Encoding a test instance's numeric class value with respect to a machine learning scheme requires the error between the scheme's prediction and the actual value be encoded. This in turn requires a probability distribution over the reals that is sensitive to the machine learning scheme's predictive accuracy. We assume the scheme's prediction errors will be normally distributed, which can be described by the mean and standard deviation of the scheme's prediction errors as testing progresses.

Let the current test instance a have numeric class value x. Let the predicted class value for a be $f(a)$. Let the mean and standard deviation of the predictions thus far be μ and σ, respectively. The resulting probability density function is given by

$$P(x|a) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(f(a)-x-\mu)^2}{2\sigma^2}}.$$

A numeric value x can then be encoded in $-log_2(P(x|a)) + C$ bits, where C is a term used to specify the accuracy to which x is given.

Calculating the entropy gain measure for numeric class attributes requires two such probability distributions: one inferred from the prediction errors of the machine learning scheme of interest (denoted $P_{scheme}(x|a)$), and the other inferred from the prediction errors of a "naive" machine learning scheme (denoted $P_{naive}(x|a)$). The naive method we use simply predicts the mean of the observed values.

Assuming the accuracy term C is constant for all schemes, the entropy gain measure is defined as

$$\begin{aligned} EG_{scheme} &= \sum_{i=1}^{N} \frac{entropy_{naive}(x_i|a_i) - entropy_{scheme}(x_i|a_i)}{N} \\ &= \sum_{i=1}^{N} \frac{-log_2(P_{naive}(x|a)) + C + log_2(P_{scheme}(x|a)) - C}{N}. \\ &= \sum_{i=1}^{N} \frac{-log_2(P_{naive}(x|a)) + log_2(P_{scheme}(x|a))}{N}. \end{aligned}$$

It remains to determine initial values for the standard deviation of each scheme's prediction errors (this is the numeric equivalent of the zero frequency problem). If the initial standard deviation is underestimated, there may be a large variation in the entropy gain for the first test instances that could bias the final entropy gain figures, particularly if the total number of test instances is low. The initial standard deviation we use (in the absence of other knowledge) is the absolute value of the first observed class value. Since the standard deviation calculated from the schemes predictions is unstable until a number of predictions have been made, the default standard deviation is used until 5 instances have been observed.

3 Numeric Prediction in Agricultural Datasets

3.1 Schemes

The schemes employed in these experiments were the instance-based learners IB1, K*, and the decision tree learner M5′. These are currently the only schemes in the WEKA workbench capable of predicting numeric values. IB1 finds the most similar training instance, and uses its class value as the prediction for the test instance. K* calculates the expected numeric value using the transformation probability between the test instance and each training instance. M5′ partitions the instance space using a decision tree; each partition contains a linear model to predict the numeric values.

AppleBruising	1536	1260	6	8
Eucalyptus1	642	28	6	14
Eucalyptus2	642	1	6	6
GrassGrubs1	173	23	3	2
GrassGrubs2	173	11	3	1
GrassGrubsRain1	19	0	1	5
GrassGrubsRain2	19	0	1	4
GrassGrubsRain3	19	0	1	4
PastureProduction	36	0	2	22
Squash1	52	7	2	23
Squash2	52	1	2	9

Table 1: Numeric Prediction—Dataset Characteristics

Neither of the instance-based learners employ attribute weighting—the presence of irrelevant attributes can significantly degrade their performance. In contrast to the instance-based learners, the selection of relevant attributes is central to the M5′ algorithm, both in the instance space partitioning and in constructing the linear models at the leaves. A large performance difference between instance-based learners and M5′ indicates a variance in attribute importances.

IB1 bases its prediction on the single nearest neighbour, while K* weighs each training instance proportional to its similarity. A large performance difference between these two schemes indicates the concepts being learned are not cleanly defined by a few training instances (perhaps because of noise in the data). Since M5′ produces its linear models from all training instances in each partition, its performance on these datasets will be more similar to K* than IB1.

3.2 Datasets

This section describes the agricultural datasets used in evaluation. Table 1 shows a brief summary of the dataset characteristics. The arff headers for each dataset are given in Appendix A.

AppleBruising: This dataset contains attributes related to apple bruise sizes, such as the type of surface that the apple impacted against, the size of the apple, the bruise width etc. The task is to predict the impact energy that produced the bruise.

Eucalyptus1: This dataset contains the results of trials that measured Eucalyptus tree growth and survival at various locations. The dataset contains attributes related to the tree species (species and seed lot), trial site (rainfall, altitude, etc), and measured tree characteristics (insect resistance, stem/crown/branch form). The task is to predict the overall utility of eucalyptus trees given all other attributes.

Eucalyptus2: As for Eucalyptus1, except all tree-specific attributes like branch/stem/crown form have been removed. Utility is therefore predicted only from location and species information.

GrassGrubs1: This dataset contains measured grass grub population density from a variety of locations and the level of pasture damage present. The task is to predict the damage ranking from the grass grub population and the location information (whether the site is coastal/midplain, the type of irrigation the site receives).

GrassGrubs2: As for GrassGrubs1, except damage ranking is predicted given only location info (that is, no information about grass grub density is provided).

GrassGrubsRain1: Grass grub populations have also been studied with respect to rainfall levels. Attributes in this dataset describe the rainfall levels in November, Summer, and Autumn, along with the grass grub population density and pasture damage. Note: the low number of instances (19) means that accurate entropy gain results cannot be obtained. In 10 fold cross validation, there are typically only 2 instances in the test set, so the entropy gain calculation will be using the default standard deviation values. The task with this dataset is to predict the level of pasture damage given all other attributes.

AppleBruising	**1.47**	−0.04	1.35	1.26	1.40	1.41	1.27
Eucalyptus1	0.90	0.32	1.00	0.93	1.04	1.08	**1.09**
Eucalyptus2	**0.49**	0.15	0.43	0.41	0.43	0.40	0.33
GrassGrubs1	0.14	−0.25	0.11	0.03	0.16	**0.20**	**0.20**
GrassGrubs2	0.11	−0.39	0.14	0.12	0.15	**0.16**	**0.16**
GrassGrubsRain1	**0.45**	0.14	0.27	0.25	0.27	0.30	0.32
GrassGrubsRain2	0.12	−0.01	−0.03	−0.11	0.02	0.08	**0.16**
GrassGrubsRain3	0.33	0.29	0.26	0.24	0.28	0.31	**0.34**
PastureProduction	**0.33**	0.29	0.31	0.30	0.31	0.32	0.32
Squash1	0.16	0.16	0.16	0.16	0.16	**0.17**	0.16
Squash2	**0.16**	0.15	**0.16**	**0.16**	**0.16**	0.15	0.15

Table 2: Numeric Prediction—Entropy Gain Summary

GrassGrubsRain2: As for GrassGrubsRain1, except the grass grub density attribte has been removed (that is, predict the level of pasture damage from rainfall information only).

GrassGrubsRain3: As for GrassGrubsRain1, except the pasture damage attribute has been removed, and the task is to predict grass grub density from rainfall information only.

PastureProduction: The objective with this dataset is to predict pasture production from a variety of biophysical factors (such as, paddock slope, soil porosity, number of earthworms, etc).

Squash1: The task with this dataset is to predict squash fruit acceptability from factors such as fruit glucose levels, weight, colour, and various subjective measurements of sweetness/flavour etc. This dataset contains many predictor attributes that require the fruit be damaged in order to determine their values.

Squash2: As for Squash1, except all only non-destructive attributes are used for prediction.

3.3 Results

Ten repetitions of a ten-fold cross validation were carried out for each dataset. Table 2 shows the entropy gain results for all schemes on all datasets. All three schemes used default settings. In addition, several values other than the default of 20 were tried for K*'s "blend" parameter, which varies the percentage of training instances that have influence over the final prediction. These results are shown in Figure 1. Observing the change in entropy gain over a range of blend settings gives an idea for how cleanly defined the domain concepts are. If the highest entropy gain is obtained with a low blend value, the domain concepts are very cleanly defined, if the entropy gain is maximised at a high blend setting, the domain concepts are not cleanly defined.

The first general point to note is that the higher entropy gain figures are obtained on datasets with more training instances, as is to be expected. In several datasets, IB1 (and K* with low blend values) have negative entropy gains, indicating there is little domain information available from the few nearest training instances. For some of the datasets (such as *GrassGrubs1*, *GrassGrubs2*, *GrassGrubsRain2*, *Squash1*, and *Squash2*) it appears there is very little information extracted by any of the machine learning schemes.

Of all the datasets, *AppleBruising* appears to contain the most accessible information (M5′ has an entropy gain of around 1.5 bits per instance), although little of that is contained entirely by the nearest neighbour (evidenced by IB1's negative entropy gain). *Eucalyptus1* also contains learnable information, as M5′ and K* both have entropy gain figures around 1 bit per instance. Much of the information in this dataset is contained in the measured attributes, as the entropy gain figures are approximately halved for the *Eucalyptus2* dataset.

IB1 typically performs poorly, primarily because its predictions are based on the class value of the single nearest training instance. M5′ performs better than both of the instance-based learners on most datasets, probably because M5′ carries out attribute selection, whereas all attributes are used in the instance-based learners' distance measures. As most of the datasets appear to contain little information (even by M5′s results), attribute selection (or weighting) is likely to play a more important role than for datasets where large entropy gains are achieved. This point is illustrated

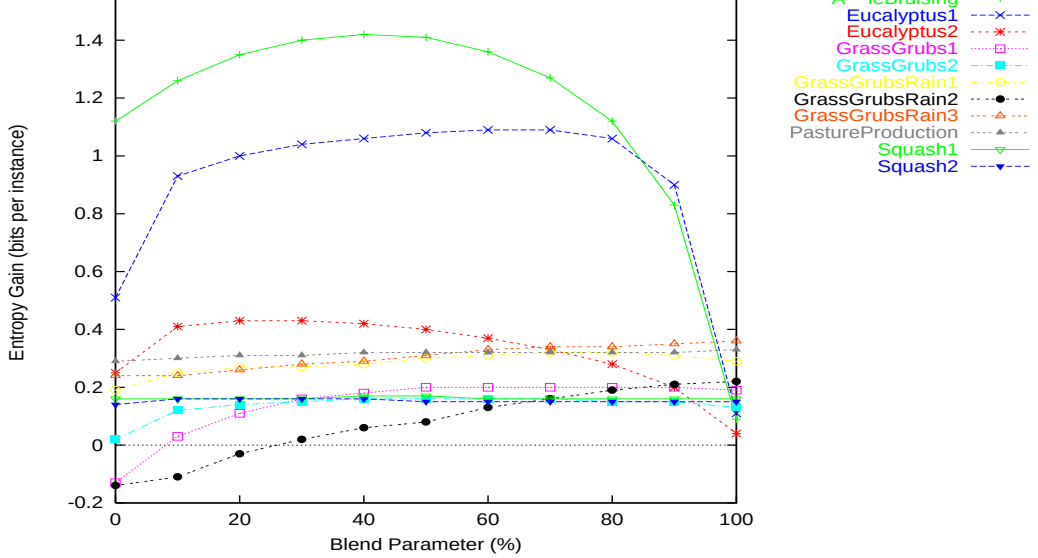

Figure 1: Entropy Gain vs K* blend parameter

with the Eucalyptus datasets. Eucalyptus1 contains eight attributes more than Eucalyptus2, all of which are relevant to the class attribute (they are all attributes that measure some aspect of tree utility). With a large number of relevant attributes, the instance-based learners perform well, but on the smaller dataset attribute selection is more crucial, so M5′ performs best.

4 Comparison of Categorical with Numeric Entropy Gain

The entropy gain measure is proposed as a common evaluation measure for both categorical and numeric predictions. In this section, we examine whether the current entropy gain measure fulfills that promise—we choose a domain where the class attribute can be fairly represented in both categorical and numeric form, and compare the entropy gain measure for both class representations. To avoid introducing biases caused by using different machine learning schemes for the categorical and numeric prediction, we use a machine learning scheme (K*) that is able to produce probability distributions for categorical class attributes as well as making numeric predictions.

Clearly, the problem domain cannot have more than two classes—representing three classes as a numeric attribute will place ordering information on the problem that will cause biases (for example, class '1' will be more similar to class '2' than class '3'). For this experiment we use a two-class problem with eight predictor attributes: two attributes are good predictors of the class; the remaining six are irrelevant. The numeric version of the domain simply represents the classes with the values 0 and 1. For any two-class problem that is very easily predicted, we would expect an entropy gain not much below 1 bit per instance.

Since the entropy gain measure adapts during testing, we vary the number of test instances, to examine whether the adaptive procedure gives stable results as the number of test instances changes. For each trial, ten training instances and a number of test instances are randomly generated, for both domain representations. Ten trials are carried out for each number of test instances. The results are shown in Figure 2.

The difference between categorical and numeric entropy gain results are markedly different for the dataset without irrelevant attributes. The categorical entropy gain figures remain around 0.74 bits per instance (bpi), regardless of the number of test instances. The numeric entropy gain figures are considerably higher (around 6 bpi on average), but with a great deal of variation. Instinctively, it should be impossible to achieve an entropy gain higher than 1 bpi, since without any learning, only 1 bit is needed to specify which of the two class values the test instance belongs to. However, the

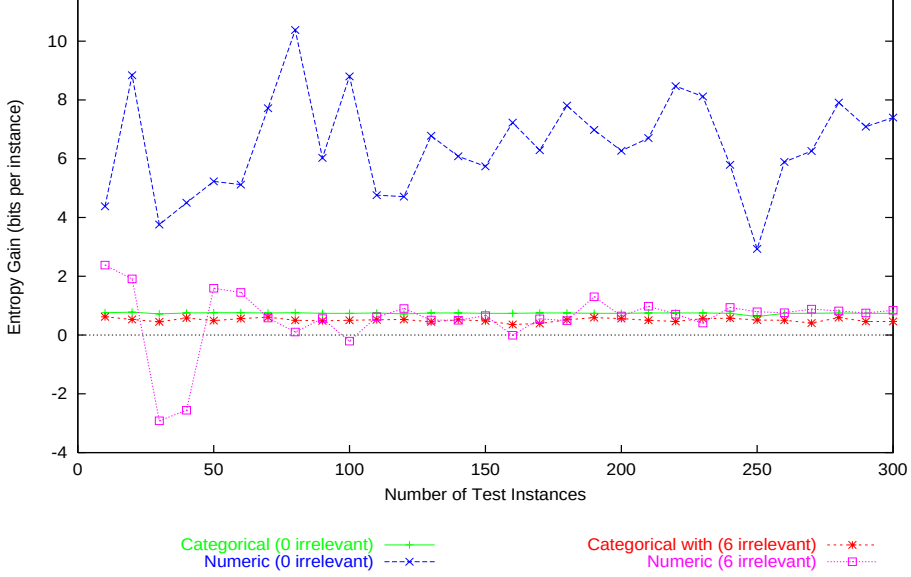

Figure 2: Comparison of Categorical and Numeric Entropy Gain

numeric predictions are so accurate that the standard deviation of the errors becomes smaller than the accuracy to which predictions are supposedly being made. This causes both the misleadingly high entropy gain figures as well as the sensitivity to errors evident in the large entropy gain variation.

The results for the dataset containing irrelevant attributes confirms that this is the case. Since classification is harder for this dataset, the standard deviation of the numeric prediction errors does not approach zero—the entropy gain in this case averages 0.71 bpi. The variation in entropy gain is initially large but stabilises after 60 test instances. The entropy gain for categorical class prediction on this dataset is around 0.52 bpi. In this more difficult dataset the two entropy gain measures give more similar results, although the difference between them is still large. For example, a further problem with this dataset is that the assumption of normally distributed errors is violated. With this dataset, the distribution of errors is distinctly bimodal, with peaks at -1 and 1.

This simple experiment shows that the current entropy gain measures are not entirely comparable. The disparity is because we are trying to infer a probability distribution over the reals based solely on numeric predictions and their errors, rather than being given a probability distribution by the scheme directly.

For the time being, the entropy gain measures are useful on "real" datasets where the learning task is likely to be non-trivial, where the variation in predicted numeric attributes ensures the estimated error standard deviation is not smaller than the attribute precision. In these datasets, the assumption of normally distributed errors is also less likely to be violated.

5 Conclusions

The entropy gain measure is useful with current machine learning scheme implementations, particularly on real world datasets. However, these measures could be improved if the machine learning schemes were developed to produce probability distributions over the class attribute.

Essentially, both categorical and numeric entropy gain measures have problems introduced because current machine learning schemes fail to return enough information. These problems work against the accuracy of the entropy gain measure. If categorical classifiers returned a probability distribution, and numeric classifiers returned a distribution over the reals (or at least a predicted standard deviation as well as the predicted value, if we keep the assumption of normally distributed errors),

machine learning schemes, and is an area for further work in the future.

6 References

Cleary, J., S. Legg and I.H. Witten (1996) "An MDL estimate of the Significance of Rules." *Proceedings of the Information, Statistics and Induction in Science Conference*, pp. 43–53, Melbourne, Australia.

Wittend, I.H. and T.C. Bell (1991) "The Zero-frequency problem: estimating the probabilities of novel events in adaptive text compression." *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1085–94.

7 Appendix A

AppleBruising Dataset Header

```
@relation apples
@attribute Code { 95GS-A, 95GS-B, 95GS-C, 95GS-D}
@attribute HarvestTime { Mid}
@attribute ImpactSurface { Steel, Fruit}
@attribute ImpactEnergy real
@attribute FruitNumber { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}
@attribute BruiseLabel { A, B, C, D}
@attribute FruitRadiusCombined integer
@attribute ContactWidthEquator real
@attribute ContactWidthPolar real
@attribute BruiseWidthEquator real
@attribute BruiseWidthPolar real
@attribute BruiseDepthBottom real
@attribute BruiseDepthTop real
@attribute VisibleExternally { 0, 1}
```

Eucalyptus1 Dataset Header

```
@relation eucdat
@attribute Abbrev { Cra, Cly, Nga, Wai, K81, Wak, K82, WSp, K83, Lon,
Puk, Paw, K81a, Mor, Wen, WSh}
@attribute Rep integer
@attribute Locality { Central_Hawkes_Bay, Northern_Hawkes_Bay,
Southern_Hawkes_Bay, Central_Hawkes_Bay_(coastal), Central_Wairarapa,
South_Wairarapa, Southern_Hawkes_Bay_(coastal), Central_Poverty_Bay}
@attribute Map_Ref { N135_382/137, N116_848/985, N145_874/586, N142_377/957,
N158_344/626, N162_081/300, N158_343/625, N151_912/221, N162_097/424,
N166_063/197, N146_273/737, N141_295/063, N98_539/567, N151_922/226}
@attribute Latitude { 39__38, 39__00, 40__11, 39__50, 40__57, 41__12, 40__36,
41__08, 41__16, 40__00, 39__43, 82__32}
@attribute Altitude integer
@attribute Rainfall integer
@attribute Frosts integer
@attribute Year integer
@attribute Sp { co, fr, ma, nd, ni, ob, ov, pu, rd, si, mn, ag, bxs, br, el,
fa, jo, ka, re, sm, ro, nc, am, cr, pa, ra, te}
@attribute PMCno { 1520, 1487, 1362, 1596, 2088, 1522, 1521, 1523, 1524, 1525,
1094, 1111, 1112, 1113, 1338, 1482, 1978, 1339, 1786, 2447, 1595, 1171,
2459, 1785, 1598, 1787, 2482, 2431, 1340, 2428, 1337, 1192, 2435, 2469,
```

```
2550, 2575, 1592, 2560, 2569, 2553, 2574, 2638, 1788, 2571, 2562, 1252,
2547, 2568, 2570, 2750, 2458, 1, 2751, 1352, 2432, 2622}
@attribute DBH real
@attribute Ht real
@attribute Surv integer
@attribute Vig real
@attribute Ins_res real
@attribute Stem_Fm real
@attribute Crown_Fm real
@attribute Brnch_Fm real
@attribute Utility real
```

Eucalyptus2 Dataset Header

```
@relation eucdat
@attribute Abbrev { Cra, Cly, Nga, Wai, K81, Wak, K82, WSp, K83, Lon, Puk,
Paw, K81a, Mor, Wen, WSh}
@attribute Rep integer
@attribute Locality { Central_Hawkes_Bay, Northern_Hawkes_Bay,
Southern_Hawkes_Bay, Central_Hawkes_Bay_(coastal), Central_Wairarapa,
South_Wairarapa, Southern_Hawkes_Bay_(coastal), Central_Poverty_Bay}
@attribute Map_Ref { N135_382/137, N116_848/985, N145_874/586, N142_377/957,
N158_344/626, N162_081/300, N158_343/625, N151_912/221, N162_097/424,
N166_063/197, N146_273/737, N141_295/063, N98_539/567, N151_922/226}
@attribute Latitude { 39__38, 39__00, 40__11, 39__50, 40__57, 41__12, 40__36,
41__08, 41__16, 40__00, 39__43, 82__32}
@attribute Altitude integer
@attribute Rainfall integer
@attribute Frosts integer
@attribute Year integer
@attribute Sp { co, fr, ma, nd, ni, ob, ov, pu, rd, si, mn, ag, bxs, br, el,
fa, jo, ka, re, sm, ro, nc, am, cr, pa, ra, te}
@attribute PMCno { 1520, 1487, 1362, 1596, 2088, 1522, 1521, 1523, 1524, 1525,
1094, 1111, 1112, 1113, 1338, 1482, 1978, 1339, 1786, 2447, 1595, 1171,
2459, 1785, 1598, 1787, 2482, 2431, 1340, 2428, 1337, 1192, 2435, 2469,
1780, 2430, 2620, 2621, 2426, 2548, 2552, 2631, 2634, 2635, 1259, 2573,
2550, 2575, 1592, 2560, 2569, 2553, 2574, 2638, 1788, 2571, 2562, 1252,
2547, 2568, 2570, 2750, 2458, 1, 2751, 1352, 2432, 2622}
@attribute Utility real
```

GrassGrubs1 Dataset Header

```
@relation Grass_grubs_and_damage_ranking
@attribute Year { 86, 87, 88, 89, 90, 91, 92}
@attribute damage_rankALL integer
@attribute GG_per_m2 integer
@attribute dry_or_irr { D, O, B}
@attribute zone { F, M, C}
```

GrassGrubs2 Dataset Header

```
@relation Grass_grubs_and_damage_ranking
@attribute Year { 86, 87, 88, 89, 90, 91, 92}
@attribute damage_rankALL integer
@attribute dry_or_irr { D, O, B}
@attribute zone { F, M, C}
```

```
@relation AUSTDAT
@attribute Year { 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983,
1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992}
@attribute GG_per_m2 real
@attribute Pct_Damage real
@attribute Nov_Rain real
@attribute Summer_Rain real
@attribute Autumn_Rain real
```

GrassGrubsRain2 Dataset Header

```
@relation AUSTDAT
@attribute Year { 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983,
1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992}
@attribute Pct_Damage real
@attribute Nov_Rain real
@attribute Summer_Rain real
@attribute Autumn_Rain real
```

GrassGrubsRain3 Dataset Header

```
@relation AUSTDAT
@attribute Year { 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983,
1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992}
@attribute GG_per_m2 real
@attribute Nov_Rain real
@attribute Summer_Rain real
@attribute Autumn_Rain real
```

PastureProduction Dataset Header

```
@relation 'pasture-production'
@attribute 'paddock-id' { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36}
@attribute 'pasture-prod-num' integer
@attribute 'fertiliser' { LL, LN, HN, HH}
@attribute 'slope' integer
@attribute 'aspect-dev-NW' integer
@attribute 'OlsenP' integer
@attribute 'MinN' integer
@attribute 'TS' integer
@attribute 'Ca-Mg' real
@attribute 'LOM' real
@attribute 'NFIX-mean' real
@attribute 'Eworms-main-3' real
@attribute 'Eworms-No-species' integer
@attribute 'KUnSat' real
@attribute 'OM' real
@attribute 'Air-Perm' real
@attribute 'Porosity' real
@attribute 'HFRG-pct-mean' real
@attribute 'legume-yield' real
@attribute 'OSPP-pct-mean' real
@attribute 'Jan-Mar-mean-TDR' real
```

```
@attribute 'root-surface-area' real
@attribute 'Leaf-P' real
```

Squash1 Dataset Header

```
@relation trainst
@attribute site { P, HB, LINC}
@attribute daf integer
@attribute fruit { 1, 2, 9, 10, 7, 11, 17, 3, 4, 12, 8, 13, 5, 15,
6, 20, 14, 23, 27, 16, 19, 21}
@attribute weight real
@attribute storewt real
@attribute pene integer
@attribute solids_% integer
@attribute brix integer
@attribute a* integer
@attribute egdd real
@attribute fgdd real
@attribute groundspot_a* integer
@attribute glucose integer
@attribute fructose integer
@attribute sucrose integer
@attribute total integer
@attribute glucose+fructose integer
@attribute starch integer
@attribute sweetness integer
@attribute flavour integer
@attribute dry/moist integer
@attribute fibre integer
@attribute acceptability integer
@attribute heat_input_emerg real
@attribute heat_input_flower real
```

Squash2 Dataset Header

```
@relation trainst
@attribute site { P, HB, LINC}
@attribute daf integer
@attribute fruit { 1, 2, 9, 10, 7, 11, 17, 3, 4, 12, 8, 13, 5, 15, 6,
20, 14, 23, 27, 16, 19, 21}
@attribute weight real
@attribute storewt real
@attribute egdd real
@attribute fgdd real
@attribute groundspot_a* integer
@attribute acceptability integer
@attribute heat_input_emerg real
@attribute heat_input_flower real
```