Euredit D4.4.1 and D5.4.1: Application of CMM techniques to data editing and imputation

Ken Lees, Simon O'Keefe, and Jim Austin
Advanced Computer Architectures Group
Department of Computer Science
University of York

This page left intentionally blank.

Contents

Summary

This report describes the general approach taken in applying Correlation Matrix Memory (CMM) neural network methods to data editing and imputation and represents deliverables D4.4.1 and D5.4.1 for the Euredit project. This report also provides an update on edit results and experiments performed since an earlier version of Deliverable D4.4.1 was submitted at month 24 (at which time evaluation results were unavailable).

CMM is a type of neural network that is "trained" to associate pairs of patterns. Most neural networks require many training cycles through training data, but CMM only requires a single pass through the training data to learn an association. Research at University of York is focused on a special binary version of CMM that uses only binary elements in the input and output patterns, and in the weights stored by the network. Binary CMM can be implemented very efficiently and the result is a very fast, scalable, pattern matching method that can deal with large data sets. Applications typically use the CMM as a kind of *filter* to remove patterns that do not match closely with the input pattern, so that a conventional (but possibly slow) algorithm can be applied to the relatively small number of remaining patterns. It is important to understand that CMM is trained to represent explicit features of the data, whereas most neural networks are trained to represent implicit features of some assumed model, which to some degree "explains" the data in a training set. This means that it is not necessary to provide a completely "clean" training set with the CMM method since no implicit model assumptions are made.

The use of CMM for error localisation and imputation involves pre-processing the data to obtain a suitable binary pattern representation. The CMM is used to find a set of best matches for each data record. This best match set is then used to calculate Euclidean distances from the record to each match within this neighbourhood.

For imputation, the best match set represents the empirical distribution in a local region in n-space, near to a record having missing values to be imputed. This set is used as the basis of a chosen imputation mode and currently five main modes are available: nearest neighbour, random neighbour, mean, weighted mean, and median. Each mode thus prescribes the method for selecting a suitable value for imputation from the local set. For error localisation, we compute a measure of Euclidean distance for each record to its k^{th} neighbour (for a suitable preset value of k). Essentially, the larger this distance is, the more likely it is that the record in question is an outlier. We refer to this process as the **DKN** ("Distance to K^{th} Neighbour") strategy.

It is not possible to derive many general conclusions about the results presented here because they are based on the use of a single basic approach – CMM. Later during WP6 of the Euredit project these CMM results will be compared with the results obtained in other work packages using different methods, and then a clearer picture of performance in relative terms should be become available. For error localisation, it is apparent that there is some trade-off in which the cost of improving the error detection rate is an increase in the false-alarm rate. For imputation, the *nearest-neighbour* mode often performs good-to-best for preservation of distribution according to the Kolmogorov-Smirnov criteria, whereas the *weighted-mean* mode often performs good-to-best for preservation of values (criteria dL1, dL2, and dLinf).

In working towards the objective of automatic edit and imputation, the York system has been developed as much as possible to minimise the need for effort or knowledge on the part of an end-user, with most system parameters having suitable default values.

1 Introduction

1.1 Introduction

This document describes the general approach taken in applying Correlation Matrix Memory neural network methods to data editing and imputation. In the Euredit project, the term "editing" was taken to mean "error localisation" from the outset, and these terms are used interchangeably throughout this report. Specifically, the techniques described here are intended to address only that part of data editing concerned with identifying errors or outliers.

1.2 Background

Correlation Matrix Memory (CMM) is a type of neural network that is "trained" to associate pairs of patterns (comprising an input pattern and an output pattern). Most neural networks require many training cycles or passes through a training pattern data, but CMM only requires a single pass through the training data to learn an association. Later, if the input pattern only is presented to the network, the corresponding output pattern is "recalled" (reproduced at the networks' output). Additionally, if the input pattern presented is incomplete or contains errors, the output will typically comprise a number of "recalled" patterns representing partial-matches with the corrupted input pattern.

At University of York, a special binary version of CMM is under study and development. Binary CMM uses only binary elements in the input and output patterns, and in the weights stored by the network (unlike many neural networks which use real-valued elements in the input and output patterns and weights). In addition to the benefits described above, binary CMM can be implemented very efficiently in software (with supporting, dedicated hardware if required). The result is a very fast, scalable, pattern matching method which can deal with large data sets which may be beyond the practical scope of some other neural network methods.

A typical approach taken with CMM applications involves using the CMM as a kind of *filter* to remove patterns that do not match closely with the input pattern, leaving just a smaller number of patterns that may be of interest. Then, a more conventional algorithm can be applied to the relatively small number of remaining patterns. Often, it is the case that using a conventional algorithm on the full data set would produce the desired results, but would take an unacceptably long time to process. Using a CMM first as a filter, the effect is to produce a smaller data set on which a conventional algorithm can produce the desired results in a more acceptable time.

CMM forms a central part of AURA (Advanced Uncertain Reasoning Architecture). AURA is a family of techniques developed for the construction of high-speed pattern matching systems with a wide range of real-world applications. AURA and CMM have been studied at University of York for over 15 years.

AURA operation is different from most neural networks (though fundamental similarities remain). The AURA approach involves storing and comparing large numbers of features selected from the data. The core CMM component of AURA is based on a simple one-layer neural network that uses binary weights and Hebbian learning, with origins in the Learning Matrix (Steinbuch, 1961). Fundamentally, the memory associates an input pattern with an output pattern. For computational efficiency, a binary version of CMM is used that has binary weights and inputs. An important feature of CMM is a powerful partial-match mechanism, that supports search and match operations with incomplete and corrupt data either in the

unknown query or in the stored patterns. High-performance is achieved through the use of fast computations with binary-valued weights and states, together with "one-shot" training. Patterns are stored using a few simple logical operations. In contrast, most other neural networks are associated with long training times. The use of CMM drastically reduces the computational problem of managing large numbers of features in the training examples.

AURA is currently implemented as a C++ class library for a range of platforms, and implementations using conventional digital hardware have been demonstrated (Austin, Kennedy, 1998).

An earlier smaller-scale project at University of York investigated the use of CMM methods for imputation under contract to Eurostat (reference 8223008/SUP-COM) and is available as a Euredit report (Austin and O'Keefe, 1999).

1.3 Approach in outline

In applying CMM methods to editing and data imputation, the AURA software library is supplemented with functions for quantisation of real-valued data developed in a different project, with functions for finding the k nearest-neighbours developed in another project, and with functions developed specifically within the Euredit project. The latter perform actions specific to the needs of Euredit Work Packages 4.4 and 5.4 and adapt the behaviour of other functions to the differing requirements of data sources in Euredit.

When used in Euredit for edit and imputation applications, CMM can be viewed as a highly flexible type of index system. For a given dataset, we first decide on the size and structure of the CMM according to the amount and types of data concerned. Next we train the CMM by setting certain bits to '1', representing the presence of a particular value or range of values in each record. In this case, the trained CMM represents a mapping from possible data values to each individual record in the data file that contains those values. At this point, we may take another (possibly previously unseen) data record and, using the CMM, determine which other records are similar to this *query* record. In particular, if this data record has some missing values, the CMM will identify matches using only the non-missing values.

The matching records produced by the CMM are identified in no particular order, but it is assumed that these matches lie within a region of space around the *query* record, which is approximately Euclidean in character (in practice, this assumption is not always correct – see section 1.4 below).

Briefly, the use of CMM for error localisation involves pre-processing the data to obtain a suitable binary pattern representation for the CMM. After storing the binary pattern representations for all data records in a CMM using "train mode", the CMM is used in "recall mode" to find a set of best matches for each record. This best match set is then used to calculate Euclidean distances from the record to each match within this neighbourhood, leading to a measure of Euclidean distance for each record to its k^{th} neighbour (for a suitable preset value of k). Essentially, the larger this distance is, the more likely it is that the record in question is an outlier. We refer to this process as the DKN ("Distance to K^{th} Neighbour") strategy.

Similarly, the use of CMM for data imputation involves pre-processing the data to obtain a suitable binary pattern representation for the CMM. After storing the binary pattern representations for all data records in a CMM using "train mode", the CMM is used in "recall mode" to find a set of best matches for each record. This best match set is then used to calculate Euclidean distances from the record to each match within this neighbourhood. From this point, the best match set may be analysed further in a number of ways to select a suitable value to be imputed in place of the missing item. In the experiments described here five main

"modes" of selecting a suitable value are considered: nearest-neighbour, random neighbour, median, mean and weighted-mean. These imputation "modes" are described in more detail in section 3.3.

The key steps in the process are shown in Table 1.

1. Configure the system with suitable pre-processing from external data to binary patterns
2. Train CMM using a binary pattern representation of every data record
3. For each record **P**:
   - 3.1. Perform a recall from CMM using binary pattern representation of **P** as a *query*
   - 3.2. Find the j best-matches, where j > k
     (since CMM matching can only *approximate* Euclidean-distance based matching)
   - 3.3. Compute the k-NN subset from the j best-matches as follows:
     - 3.3.1. Retrieve values from data file for every matching neighbour record
     - 3.3.2. Find Euclidean distance from **P** to each matching neighbour record
     - 3.3.3. Sort the matching neighbours according to distance to find true k-NN subset

Then either:

4. Store **DKN** – the distance from record **P** to it's k^{th} neighbour (value of k is preset – see below) in a table **T**
5. Sort table **T** according to distance to obtain a ranked list (showing for every record the distance to that records' k^{th} neighbour)
6. Determine a threshold cut-off distance based on the user-supplied parameter **sd**. Distances greater than **sd** indicate an error, while a distance equal to **sd** or less indicates an acceptable record. The error-status of each value in an error record is computed based on individual contribution to the **DKN** value.

Or:

4. Impute a suitable replacement for each missing value in **P,** using the user-selected mode.

Table 1: Key steps in the use of CMM for edit and imputation

1.4 Improvements to CMM matching

It was originally assumed that the binary matching process associated with CMM provides a "reasonable" approximation to the comparison of Euclidean distances. Thus, as long as it is ensured that the number of matches identified by the CMM includes a "safety" margin, it seemed reasonable to expect that the k nearest neighbours would be contained within a subset of the CMM matches. At a rather late stage, it was discovered (Hodge et al, 2002) that the original method being used was not always successful in identifying the true k-NN subset, and more distant points were sometimes included instead of nearer points. This resulted in significant further investigation and development work to better understand the reasons for this and to remedy the situation. This is discussed in more detail later in section 2.4.6 but, essentially, substantial changes were required to the pre-processing stage.

1.5 Performance and quality issues

It is important to note that virtually all effort at York during the Euredit project has been focused on the development of additional software that implements edit and imputation techniques, around the core of the existing CMM software library. Thus, although the CMM software library is now a mature and efficient implementation the same is not true of the additional software produced during the project, which is necessarily of prototype quality only. In practice, this means that although fully functional, the system evaluated in the Euredit project does not necessarily use system resources as efficiently as one might expect from production-quality software in terms of memory use and speed of computation. The core CMM software is very efficient in speed terms but the system developed for Euredit does not always reflect this efficiency for the reasons given above.

1.6 General approach

Throughout the Euredit project, work at York has adopted a philosophy of following as closely as possible the spirit of the project as a whole, especially in terms of working towards an automatic edit and imputation system. In particular, the York system has been developed as much as possible to minimise the need for effort or knowledge on the part of an end-user. The York system requires little knowledge of the actual data, and needs only the preparation of a specification file which describes the basic elements of the data to be processed (such information would normally be found in the metadata accompanying a dataset).

2 CMM

2.1 Description of CMM

CMM can be viewed in terms of a correlation matrix *M,* comprising an array of binary elements, initially set to zero. The matrix is trained according to the values of binary input and output vectors, by computing an outer product between each input vector Q_i and output vector R_i. The result is bitwise-logically ORed with the existing matrix resulting in the following update equation:

$$M^k = M^{k-1} \cup \left(Q_i \times R_i^T\right) \quad 1$$

Here, M^k is the updated correlation (or weights) matrix after k pairs of input-output patterns have been trained, and R^T is the transpose of column vector R. To prevent the memory becoming saturated, the input and output vectors are chosen to have a small, fixed bit density. To find the set of nearest-matching stored patterns, a recall operation is performed. The inner-product of the unknown input pattern with the matrix M is computed, forming integer-valued elements in an output vector G:

$$G_i^T = I_i^T M^k \quad 2$$

For the work described here, a threshold function known as Willshaw thresholding (Willshaw et al, 1969) is used. Willshaw thresholding operates by setting a '1' in the final binary output when the corresponding element in the output vector G contains a value greater than or equal to a threshold value θ determined as the number of bits set to '1' in the input pattern (I_i^T in equation 2) presented during recall, since this represents the maximum response of any element in the output vector G. As a result, elements of G with values below the threshold are set to zero, and all others are set to one.

In summary, the final (binary) output vector R is obtained using:

$$R^T = f_{thresh}\left(G^T\right) \quad 3$$

where:

$$f_{thresh}\left(g_i\right) = \left\{ \begin{matrix} 1, \left(g \ \geq \theta\right) \\ 0 \end{matrix} \right\} \quad 4$$

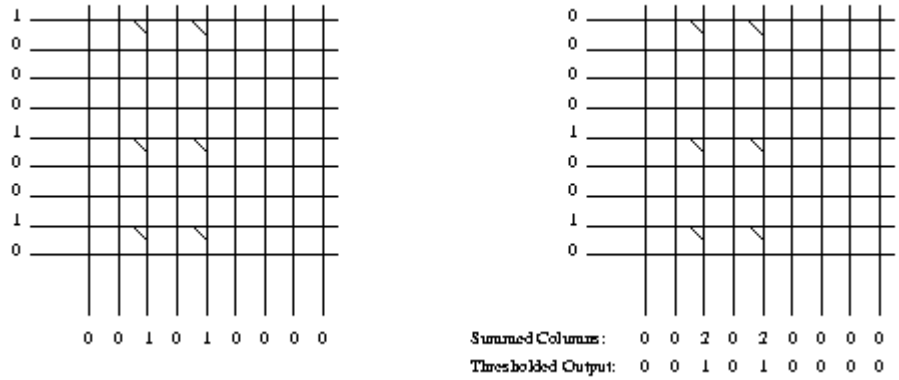
and θ is a positive integer representing the number of bits set to '1' in the input pattern. Further details of both training and recalling with CMMs can be found in (Austin, 1996).

2.2 CMM operation

The AURA techniques provide methods to pre-process external data into a suitable binary pattern form for storage in a CMM. A simplified practical explanation of basic CMM operation is given below.

2.2.1 CMM Training

In Figure 2.2.1(a) below, an input binary pattern (left vertical) is stored in the CMM by "associating" it with another "separator" pattern (left horizontal). The association is formed by making appropriate connections between the rows and columns that make up the grid. A simple (Hebbian) rule determines that a connection is made at the intersection between a row and a column, when a binary '1' appears in the corresponding positions of the two patterns to be associated. Thus if the input binary pattern contains two binary '1' values and the associated "separator" pattern also contains two binary '1' values, then four '1's will be added to the CMM. New associations between pattern-pairs are *superimposed* with existing associations, so that existing connections remain unchanged once created. An example of a very simple CMM is shown below where only a single pair of patterns is stored. Note that most "real" CMMs typically have thousands of rows and columns, equivalent to several Megabytes of conventional storage.



a) pattern storage **b) pattern retrieval**

Figure 2.2.1: Correlation Matrix Memory

2.2.2 CMM Recall

Figure 2.2.1(b) above shows how the stored association is retrieved by applying only the input binary pattern (left vertical). Wherever a '1' appearing in the input pattern coincides with a connection in the corresponding row of the CMM, a count is incremented for the column connected to that row. The intermediate result produced comprises an array of raw output summations over the CMM columns (right horizontal) and is labelled "Summed Columns" in Figure 2.2.1(b).

A single, global threshold is applied to the output summation values to obtain the binary "separator" pattern originally associated with the input pattern. Several methods exits for choosing a suitable threshold value, but a useful basic method is to simply count the number of rows containing a '1' value in the input pattern and use this as the threshold. In the example above the number of '1' values is 2 and this is the threshold value used.

The "Thresholded Output" shown in Figure 2.2.1(b) is the results of applying a threshold, and is a binary pattern vector termed the "separator". In general, the separator may represent the superposition of a number

of binary pattern vectors (each corresponding to a different match). Other AURA techniques are used to recover the individual patterns, translating these into the actual matching data items.

2.3 Finding the k-nearest neighbours using CMM

CMM can be used to implement a k-nearest neighbour (k-NN) approach to finding outliers for data editing applications. Some related work – the implementation of a classifier based on k-NN using AURA technology – was described in (Zhou et al, 1999).

Although it would be straightforward to determine the k-NN in a dataset directly, this is computationally expensive for large datasets. Instead, a CMM neural network (within the AURA framework) is used to find a smaller subset of records containing the k-NN, and then to apply the k-NN rule to the subset. The subset found by a CMM always contains the required data, together with a few unwanted records (Turner et al, 1997). The latter are removed when the k-NN rule is applied to the subset. The scalable, high-speed pattern matching technology provided by AURA means that the k-NN are found very quickly by this indirect method. A detailed description of the implementation of a CMM k-NN classifier is given in (Zhou et al, 1999).

For Euredit, instead of using the k-NN method for direct classification, we use the k-NN as a localised sample (in Euclidean-distance terms) of the larger data space represented by the full dataset. This local neighbourhood sample contains much useful information about properties such as the local data density. Local data density can provide a useful heuristic guide to the presence of novelty or outliers in the data.

Pre-processing is needed to extract suitable features from datasets for training in the CMM neural network. Essentially, this involves assigning appropriate binary patterns for each value in an input data record, and then combining these patterns to form a complete input pattern to the network. To simplify the mapping of continuous variables into binary patterns, continuous values are quantised using Robust Uniform Encoding (RUE), which is detailed in (Zhou et al, 1999). RUE is optimal in the sense that the number of continuous values assigned to each bin is approximately constant for a given training set. This is a very desirable property because evenly distributed training patterns make best use of CMM storage by avoiding saturated regions which could otherwise result in excessive false matches during recall. Finally, a binary pattern must be allocated for each quantised bin, so that continuous input-space distances are approximated within a local region by corresponding distances in the binary patterns generated. Categorical data is encoded using a distinct binary pattern for each nominal category. Ordinal categories are also encoded using a distinct binary pattern, but using a process that partially preserves ordinality.

2.4 CMM Pre-processing

As mentioned previously, AURA techniques provide methods to pre-process external data into a suitable binary pattern form for storage in a CMM. Some of methods are integrated with the AURA software library already but other methods are still to be incorporated and are currently implemented in separate functional modules. Three main variable types are considered here: categorical, continuous, and discrete ordinal types. In addition, the method is implemented so that certain variables can be ignored completely during processing, and others can be ignored by the CMM processing but considered during the post-processing stages where Euclidean distances are calculated.

Note that the pre-processing differs slightly between the training phase and the recall phase. During training, sparse binary input vectors are created which, for each variable, have just one bit set. During recall where

edit and imputation processing is performed, it is sometimes preferable to set a number of adjacent bits in the binary input vectors to better approximate Euclidean distances in the data. This is discussed in more detail in the sections below.

2.4.1 CMM input configuration issues (superposition vs. concatenation)

Before pre-processing is considered, a basic decision about configuration of the CMM inputs is needed. For some applications, it is most efficient to separately encode each part (e.g. variable, attribute, etc.) of a single input as a set of binary vectors having the same length, and then to superimpose these vectors using a bitwise logical OR operation. This is effective in cases where all input variables are very similar in type and cardinality (e.g. all continuous variables, or all categorical over a similar range). It is very efficient in memory terms, but at the cost of a loss of information whenever two corresponding vector elements are set to '1' (because these are combined in the logical OR operation).

When dealing with a range of data types as found in most edit and imputation applications, a different configuration that offers additional advantages is used. In this case, each part (e.g. variable, attribute, etc.) of a single input is separately encodes as a set of binary vectors (as before), this time having different lengths. These varying length vectors are now concatenated (i.e. end-to-end) rather than being superimposed as describes above. This implies a greater storage requirement than the superposition method described above, but allows more flexibility through the use of variable length vectors, and better preservation of input feature information.

In the Euredit project, a concatenated input configuration has been used.

2.4.2 Pre-processing for categorical data types

Categorical data is the simplest type of data from the viewpoint of CMM pre-processing and is encoded directly into a unique sparse binary vector, for each unique category present in the data, for example:

house type			
detached	semi-detached	bungalow	apartment
0001	0010	0100	1000

Figure 2.4.2: Example encoding for categorical values

2.4.3 Pre-processing for continuous data types

For each continuous variable, the data is first quantised into discrete intervals or "bins" using the RUE algorithm (Zhou et al,1999). Finally the encoding process converts the corresponding bin values into a sparse binary vector which forms the CMM input. The RUE algorithm first scans all the values for each variable in the dataset to find the optimal position for bin boundaries, so that the variation in number of sample values allocated to each bin is minimised. This has three main benefits: it avoids problems due to "empty" bins, it distributes binary patterns more uniformly in the CMM, and it provides higher resolution for highly-populated regions of the input data distribution (and vice-versa).

Although this method appears to work well, there is a risk during the recall that data points lying near a boundary will be allocated to a certain bin, when in fact many near neighbours lie in the adjacent bin. To

avoid this problem, initial experiments aimed to improve accuracy by setting bits adjacent to the bit representing the "central" bin (in which a given input data point lies). This additional step creates a binary vector containing 3 (or more if required) adjacent bits set to '1'. For example:

bin number	income	binary vector
0	<£9,500	000000000011
1	£9,500-£21,000	000000000111
2	£21,000-£28,000	000000001110
3	£28,000-£35,000	000000011100
4	£35,000-£41,000	000000111000
5	£41,000-£51,000	000001110000
6	£51,000-£62,000	000011100000
7	£62,000-£65,000	000111000000
8	£65,000-£95,000	001110000000
9	£95,000-£105,000	011100000000
10	£105,000-£150,000	111000000000
11	>£150,000	110000000000

Figure 2.4.3: Example encoding for continuous values

2.4.4 Pre-processing for discrete ordinal data types

For discrete ordinal variables, the approach taken is effectively a combination of approaches used for categorical and continuous variables, except that quantisation is no longer necessary.

First, each unique value present in the data is encoded directly into a unique sparse binary vector. Possible neighbour points in Euclidean space are included by setting bits adjacent to the bit representing the "central" input value, resulting in a binary vector containing 3 (or more) adjacent bits set to '1'.

2.4.5 Assembling the CMM input using concatenation

When all necessary pre-processing is complete, the various binary vectors representing each variable are combined using concatenation as discussed above, to form a complete CMM input. Figure 2.4.5 below shows an example of a complete CMM input encoding during recall (note that the "income" variable would have only a single bit set during training). Also, in this example, the CMM has previously been trained using a binary separator (or output vector) in which only one bit is set, so that each column of the CMM refers to a different record in the dataset.

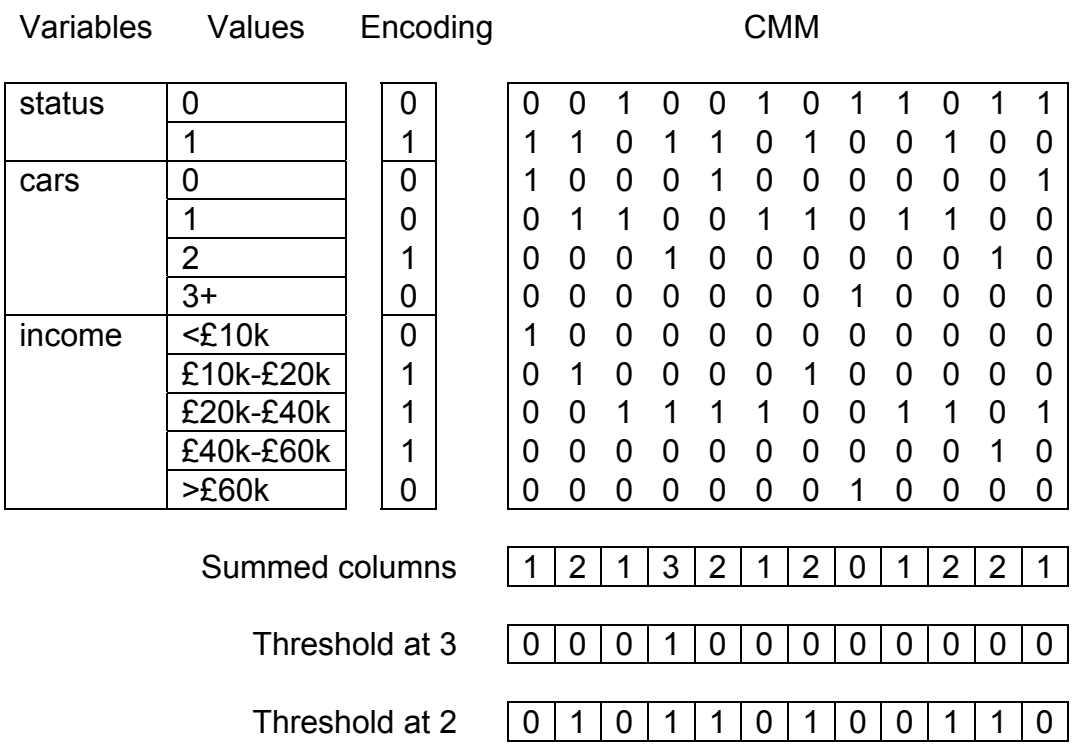


Figure 2.4.5: CMM recall with concatenated inputs

Figure 2.4.5 also shows the resulting binary output vector in the case of two different threshold values. A slight variation is used on the thresholding procedure described earlier, where an initial threshold is set at the value of the largest element in the summed columns vector, which can then be reduced incrementally to obtain the number of matches required. Thus, when a threshold value of 3 is applied, a single match is found in the 4th column, whereas when the threshold value is reduced to 2, near matches are also found in the 2nd, 5th, 7th, 10th, and 11th columns.

2.4.6 Recent enhancements to CMM pre-processing

A recent study in a different project at York (Hodge et al, 2002) indicates some undesirable properties in the current pre-processing methods applied to continuous and discrete ordinal types. It was stated above that adjacent bits in the binary input vectors are set, with the aim of better approximating the Euclidean spatial relationships in the data. However, it appears that this is not as effective as at first thought.

Although some of the neighbours generated by the CMM matching process are genuine members of the set of (Euclidean) k nearest neighbours, some others are not. The assumption was that all k nearest neighbours would be included as proper subset of the CMM matches so long as the number of CMM matches was larger than the required number k by a small margin. This is not the case with the current pre-processing methods, and appears to be due to several factors including:

1. input data is not normalised during pre-processing (though it is normalised during post processing);

2. sometimes (presumably in sparse data regions) the k nearest neighbours are not contained in the highest-ranked CMM matches – if the CMM threshold is reduced further matches are obtained but these do not approximate the required Euclidean metric as effectively as the original matches.

This second factor can be offset to some extent by setting additional adjacent bits in the binary input vectors. Adding further adjacent set bits has the effect of producing additional matches, which also approximate the Euclidean metric. However, the first factor also means that it is difficult to ensure that the Euclidean metric is applied symmetrically in each dimension. This means that some near neighbours may be included while others are not, for some values of k.

To overcome these problems a more sophisticated pre-processing scheme has been implemented. This relies on a feature of the AURA system which allows weighted input vectors to be supplied during recall so that, in addition to specifying that certain bits are set, an integer weight can also be supplied for each set bit. The first experiments used this method with a triangular weighting function centred on the value present in the query data for each continuous or discrete ordinal variable. Significant improvements in accuracy were observed (Hodge et al, 2002).

To see the form of this weighting function and how it operates, the example used previously to illustrate continuous data encoding is modified in figure 2.4.6 below to show how the triangular weighting function is applied. Clearly, a true binary vector may only take elements with the value '1' or '0' and the weighting factor is applied separately in the implementation. When this weighted binary vector is used as input to a CMM during recall, the result is that every bit set to '1' in the corresponding CMM row is effectively increased in value by the weighting factor, and this increased value is used in the accumulation of the summed columns at the CMM output.

bin number	income	"weighted" binary vector
0	<£9,500	000000000123
1	£9,500-£21,000	000000001232
2	£21,000-£28,000	000000012321
3	£28,000-£35,000	000000123210
4	£35,000-£41,000	000001232100
5	£41,000-£51,000	000012321000
6	£51,000-£62,000	000123210000
7	£62,000-£65,000	001232100000
8	£65,000-£95,000	012321000000
9	£95,000-£105,000	123210000000
10	£105,000-£150,000	232100000000
11	>£150,000	321000000000

Figure 2.4.6: Example of weighted input encoding for continuous values

2.5 The DKN strategy for error localisation

The DKN ("Distance to K^{th} Neighbour") strategy described here was devised independently at York but similar methods have been described by other authors, for example (Byers et al, 1996). The strategy starts from the rather weak assumption that the data to be analysed comprises at least two clusters, one smaller and one larger by some amount, and that there is some degree of separation between any clusters representing "outliers" and other "non-outlier" clusters. Presumably, a cluster would not be considered an "outlier cluster" unless some degree of separation is present. In general, it is expected that better performance will result from larger differences in cluster sizes and from greater cluster separation distances.

Although the data is viewed in terms of *clusters*, the DKN strategy does not aim explicitly to identify *clusters* within the data to be analysed. Instead it aims to identify *data points which are remote from other points*, but possibly sharing this remoteness with a relatively small group of neighbouring data points, and this provides indirect information about cluster characteristics present in the data.

The DKN strategy involves analysis of the *density* of data in the local neighbourhood of each point, by calculating a suitable distance metric from each point to it's nearest neighbours, in particular noting the distance to the k^{th} neighbour (the choice of parameter k is discussed later). The rationale for this approach is based on the observation that points in a sparse neighbourhood will tend to have a relatively large distance to the kth neighbour, so long as the choice of k is not too large. (If k is too large, many points will have a relatively large distance to the k^{th} neighbour.)

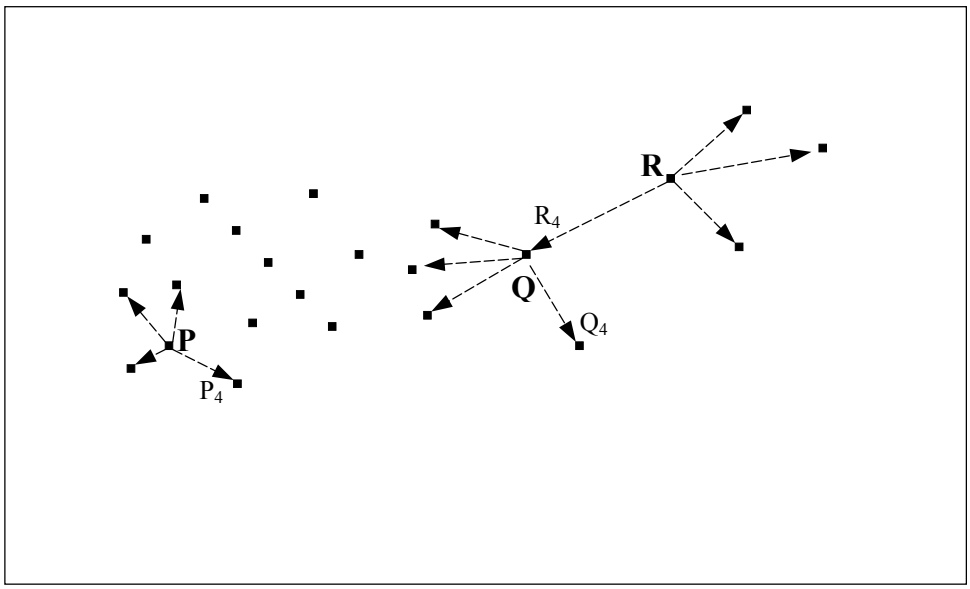


Figure 2.5: Simple illustration of the DKN approach

Figure 2.5 above shows a simple example of the DKN approach applied to a set of example points in a 2D plane. In this example, the value of k is set at 4, that is to say we are interested in finding the distance to the 4^{th} nearest neighbour of each point. For simplicity, we choose just three exemplar points P, Q, and R, to illustrate the approach.

Point P has its 4th nearest neigbour at distance P4, point Q has its 4th nearest neigbour at distance Q4, while point R has its 4th nearest neigbour at distance R4. Choosing a standard Euclidean metric would result in the ranking R4 > Q4 > P4. The inference that could be drawn from this is that point R is more likely to be in a sparse region of the data space and, ultimately, more likely to be a remote or outlying data point. Although the choice of k is clearly important it is, in general, not critical. In this example, if k=3 had been chosen, the approach would not identify point R so clearly as an outlier, since the 3^{rd} neighbour of point R is probably at a distance similar to that of the 3^{rd} neighbour of many other points in the figure. However, choosing k=5 would (as with k=4) result in a decision that R is a more likely outlier candidate.

The heuristic adopted in the experimental work described here is that the value of k should be chosen to be greater than the number of points in the largest "outlier" cluster but less than the number of points in the smallest "normal" cluster. In practice, explicit information about cluster sizes is not usually available but the number of points (records or cases) in a dataset is easily determined and, given a basic estimate of the error rate and some assumptions about outliers it is perhaps not too difficult to find a reasonable value for k.

The distance to the k^{th} neighbour is determined and stored in a distance list for every point. The distance list is then sorted in descending order so that the point with the largest value of distance appears at the head of the list. This ranked version of the list is effectively a ranking of the all points in the dataset in terms of "predicted outlyingness".

2.6 Error localisation decision threshold

When performing error localisation in practice, we need to determine a threshold or cut-off distance so that larger values of DKN indicate a record containing errors while smaller values indicate records which are judged acceptable. In the experiments described in section 4 a user-specified parameter **sd** was used to vary this threshold. The value given for this parameter is interpreted as a threshold equal to **sd** standard deviations above the overall mean DKN value, so that **sd** = 1 will set the threshold at a distance equal to the mean + 1 standard deviation.

Finally, the error-status of each value in an error record is computed in an interval between 0 and 1 (a pseudo probability) based on individual contribution to the DKN value, weighted by a score based on the actual DKN value.

3 Additional Pre- and Post-Processing

3.1 System default parameters

Appropriate default values are provided for almost all parameters used in the system so that an inexperienced user can obtain satisfactory results with a minimum of knowledge about the data to be imputed. The results described in sections 4 and 5 were produced using system defaults in almost all cases.

3.2 Valid value checks

The system checks non-real values in the input data file against user-specified ranges in the specification file to ensure these are consistent. During imputation, the system will halt and report the location of the first value in the input data file that fails this check, which must be corrected before imputation can proceed. Clearly, the availability of accurate metadata is an important factor in prescribing valid ranges in the specification file.

Note that edit rule checks are not currently supported on input or imputed value data during imputation with the system. Edit rules could be added to the system in principle, but there was insufficient time in the Euredit project to implement this feature.

3.3 Imputation modes

The system currently provides five main imputation "modes" each of which operates by selecting a suitable value from the set of k-NN neighbourhood records, as determined by the CMM and subsequent Euclidean distance calculations.

Nearest-neighbour mode simply replaces each missing value using the corresponding value copied from the neighbour that is nearest in terms of Euclidean distance.

Random-neighbour mode simply replaces each missing value using the corresponding value copied from a neighbour selected at random in the k-NN neighbourhood.

Mean mode replaces each missing value using the corresponding mean value computed over the k-NN neighbourhood.

Weighted-mean mode replaces each missing value using the corresponding value of the Euclidean distance weighted-mean, computed over the k-NN neighbourhood. This gives greater weight to values belonging to closer neighbours.

Median mode replaces each missing value using the corresponding median value computed over the k-NN neighbourhood.

Clearly other "modes" that operate effectively over a local k-NN neighbourhood could be added to this list in principle (if time was available).

3.4 Household variables

A special method was devised to ensure household variables were imputed properly for the SARs dataset. These variables are imputed in such a way as to maintain the same value for each member of the household as follows:

- "parent" and "dependents" relationships are defined in the specification file
- store the first record in each household and impute any missing values
- additional records from same household inherit household values from the first record

The current implementation of this method assumes that records for a given household are always adjacent in the input data file.

3.5 Modified Euclidean distance calculation

When a mix of categorical and continuous variables is present the calculation of Euclidean distance can become biased in favour of categorical variables if they are assigned distance values in the obvious way, i.e. using a distance of 0.0 for an exact match and a distance of 1.0 for any mismatch. Conversely, continuous variables usually have a distance value somewhere in the continuous range between 0.0 and 1.0. The net effect of this situation is to introduce a bias in which categorical variables have a larger overall effect on the calculation of distance than continuous variables.

To counter this effect, a modified calculation of distance was introduced as an option by the setting of a flag in the specification file. The modified calculation is appropriate when a mix of continuous and categorical variables is present and treats all categorical variables as a single "pseudo variable". The standard calculation for the Euclidean distance D between two records A and B (having elements a_i and b_i respectively representing the values of corresponding variables) is essentially:

$$D = \sqrt{\sum_i \left(a_i - b_i\right)^2}$$

The modified distance calculation we use is:

$$D = \sqrt{\sum_r (a_r - b_r)^2 + \left(\frac{1}{n_c}\sum_c f_B\left(a_c, b_c\right)\right)^2}$$

where:

a_r and b_r represent continuous type variables

a_c and b_c represent categorical type variables

n_c is the number of categorical variables present in each record

f is a function which returns 0 when $a_c = b_c$ and 1 otherwise

The effect of introducing the new term is to treat categorical variables collectively as a single "pseudo continuous variable" whose value ranges between 0.0 when every pair of categorical variables between the two records matches, and 1.0 when all categorical variables between the two records are different.

Indications so far suggest that this modified distance gives a better ranking of near-neighbours than the standard distance calculation resulting in better overall system performance.

3.6 Neighbourhood limit distance

With realistic data it is usual that some regions of data space are more densely populated while other regions are more sparse. By choosing to define a local neighbourhood in terms of the k nearest neighbours, it is clear that the region of space defined by this neighbourhood will vary considerably in size, according to the local population density. One effect of this variation is that the k^{th} neighbour could be quite distant when the region is sparse – possibly an extreme value which we might not wish to consider when calculating the mean (for example) of a particular variable over the neighbourhood.

A way around this difficulty is to introduce a distance limit on the local region such that, *at most k* neighbours will be considered during imputation, but neighbours must also fall within the distance limited region. This avoids the unwanted influence of extreme values in sparse regions.

The way this is currently implemented (for imputation only at present) requires a user-specified value, equivalent to the number of variables allowed *not* to match. Thus a value of 2.0 for this parameter means that a distance limit is imposed equivalent to the distance of a hypothetical neighbour record that differs in two values to the maximum degree possible (i.e. the normalised difference would be determined as 1.0 for each of two non-matching values).

3.7 Merge external error data

Although the current system at York does not support edit rules directly, an indirect method of including edit rule failure information is provided. This currently assumes a particular arrangement of error data to exploit the output of a software tool provided to the Euredit project by one of the partners (NAG). Basically, this tool provides a report for each of SARs, ABI and EPE datasets indicating the location of values which violate (so-called) hard edit rules (where there is no doubt that the value is in error). The presence of external error information is indicated by the user in the specification file, together with the name and location of the error information file. During error localisation, if the file is present, error information is taken from the external file and merged with errors found using the DKN method.

4 Error localisation results

In this section we provide an update on previous experimental results (now moved to Appendix F) with results obtained using Euredit Y3 evaluation datasets (for which the true data is held back). The procedure used during error localisation experiments involved first creating a specification file for each dataset, and creating several copies of this file. Each copy was edited so that a different value of the parameter **sd** (see section 2.6) was requested in each case. A simple DOS batch file was used to repeat the error localisation experiment for each different value (as indicated in the corresponding specification file). In most cases the system was left to choose default values, but the value of k (i.e. as required to find the DKN) was specified as 10 for all experiments reported here. This value represents a reasonable guess which is assumed to be somewhere below the lower-bound on the number of nearby records for a non-error record, and somewhere above the upper-bound on the number of nearby records for a error record. With more time, experimentation could be used to determine this value more precisely.

As always in situations like this, it is difficult to draw any firm conclusions on the basis of these results because they merely compare the effect of parameter changes but give no indication of performance relative to other methods studied in the Euredit project (relative comparisons will form the basis of WP6 in Euredit).

Three experiments were performed using different values of a threshold parameter **sd** which is tested against the **DKN** value for each record in the dataset. In all cases, the value of **k** used was 10, so that he **DKN** value computed was the distance to the 10^{th} neighbour, and a modified Euclidean distance measure was used (as described in the section 2 update).

Below is a key to the evaluation criteria names used in the tables in sections 4.1 and 4.2[1]:

Name	Meaning
alpha	estimates the probability that an incorrect value for the variable is not detected
beta	estimates the probability that a correct value is incorrectly identified as an error
delta	estimates the probability of an incorrect classification outcome for this variable
RAE	Relative Average Error
RRASE	Relative Root Average Squared Error
RER	Relative Error Range
tj	standardised measure of effectiveness in classifying this variable
AREm1	Absolute Relative Error of the k-Mean (k=1)
AREm2	Absolute Relative Error of the k-Mean (k=2)

4.1 UK Annual Business Inquiry (Retail Section)

The ABI dataset consists of a sample from the retail section of the U.K. Annual Business Inquiry (1999). For each business the dataset contains information on aspects of purchase costs and employment costs. The data

[1] See (Chambers, 2000) for more details of the performance measures used here.

is based on responses for either of 2 possible questionnaires. The larger questionnaire contains 17 retail questions. The smaller one only asks for summary information and contains 5 retail questions. Businesses on the dataset therefore either have 5 or 17 responses to individual questions. The results reported in this section are based on the Y3 version of the dataset, which contains both errors and missing values.

The first table below shows the overall performance for each value of the **sd** parameter selected. As described in (Chambers, 2000) **A** represents the proportion of records with at least one incorrect value that are passed by all edits, **B** represents the proportion of records with all correct values that are failed by at least one edit, and **C** represents the proportion of incorrect error detections. **G** is an average taken over all cases to provide an overall error localisation performance measure and a smaller value indicates better performance.

For the values of parameter **sd** chosen here the variation in performance is not very large, but there is an apparent trend for measure **A** to improve as **sd** becomes more negative between 0.0 and –0.1, while measure **B** seems to improve as the value of **sd** becomes more positive, as does **C** to a smaller extent. Measure **G** appears unchanged for the values of **sd** used here.

OVERALL	sd=0.0	sd=-0.07	sd=-0.1
G	0.024822	0.024822	0.024822
A	0.737052	0.694935	0.639158
B	0.016756	0.03731	0.06412
C	0.219798	0.222686	0.226215

Table 4.1.1: Overall measures of error localisation performance for ABI data

The tables below show the performance for each value of **sd** with each of the variables in turn TURNOVER, EMPTOTC, PURTOT, TAXTOT, ASSACQ, and ASSDISP. These variables are of primary interest as described in the Euredit evaluation handbook.

TURNOVER	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.865421	0.786916	0.695327
beta	0.025419	0.049956	0.082966
delta	0.097903	0.113548	0.135806
RAE	5.316682	3.762756	1.661336
RRASE	0.101083	0.088003	0.048657
RER	2254.207	2254.207	1536.526
tj	39.58266	28.01369	12.36864
AREm1	4.803999	3.210104	1.039363
AREm2	0.15865	0.356818	0.800202

Table 4.1.2: Error localisation results for ABI variable TURNOVER

EMPTOTC	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.836149	0.760135	0.675676
beta	0.030206	0.053262	0.085433
delta	0.107322	0.120899	0.14191
RAE	6.398992	4.687782	2.227019
RRASE	0.169812	0.151885	0.070905
RER	3733.566	3718.676	1578.132
tj	51.34303	37.61295	17.86874
AREm1	6.027577	4.256292	1.690029
AREm2	1.939754	1.403278	0.358619

Table 4.1.3: Error localisation results for ABI variable EMPTOTC

PURTOT	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.791757	0.745119	0.673536
beta	0.048891	0.070874	0.099678
delta	0.159381	0.171157	0.18503
RAE	6.548785	4.959322	2.312617
RRASE	0.137193	0.127373	0.082219
RER	4297.226	4297.226	4297.226
tj	47.21318	35.75402	16.67271
AREm1	6.348782	4.678612	1.84838
AREm2	0.518544	0.323644	0.43428

Table 4.1.4: Error localisation results for ABI variable PURTOT

TAXTOT	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.865854	0.795393	0.718157
beta	0.02619	0.048352	0.08022
delta	0.12617	0.137302	0.156179
RAE	38.20677	14.23315	6.640314
RRASE	1.961451	0.444606	0.234901
RER	61639.75	11195.08	6215.083
tj	361.8918	134.8153	62.89658
AREm1	38.36232	14.03185	6.268361
AREm2	517.73	25.98912	6.654804

Table 4.1.5: Error localisation results for ABI variable TAXTOT

ASSACQ	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.858672	0.770878	0.700214
beta	0.022464	0.04563	0.078098
delta	0.085807	0.100568	0.125223
RAE	17.86242	4.727693	1.267213
RRASE	0.979696	0.213779	0.077016
RER	32770.43	7107.048	3386
tj	90.86003	24.04816	6.445881
AREm1	17.46726	4.054431	0.464201
AREm2	36.83064	0.823234	0.759153

Table 4.1.6: Error localisation results for ABI variable ASSACQ

ASSDISP	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.832432	0.759459	0.686486
beta	0.019446	0.042161	0.071244
delta	0.068112	0.085099	0.108073
RAE	21.40873	17.17507	5.421435
RRASE	1.01857	0.768912	0.352492
RER	7452.889	3838.111	3838.111
tj	22.94304	18.40596	5.809977
AREm1	19.813	15.45976	3.318977
AREm2	0.766081	0.016506	0.783678

Table 4.1.7: Error localisation results for ABI variable ASSDISP

4.2 Swiss Environment Protection Expenditures (EPE)

The EPE dataset consists of a questionnaire distributed (in 1993) to enterprises in Switzerland chosen according to class of economic activity (usually with > 20 employees). For each enterprise the dataset contains information on expenditure relating to environmental issues. The results reported in this section are based on the Y3 version of the dataset, which contains both errors and missing values.

The first table below shows the overall performance for each value of the **sd** parameter selected and is of the same form as table 4.1.1 for the ABI data shown in the previous section. As before, **A** represents the proportion of records with at least one incorrect value that are passed by all edits, **B** represents the proportion of records with all correct values that are failed by at least one edit, and **C** represents the proportion of incorrect error detections. **G** is an average taken over all cases to provide an overall error localisation performance measure and a smaller value indicates better performance.

For the values of parameter **sd** chosen here the variation in performance is fairly small, but there is an apparent trend for measure **A** to improve as **sd** becomes more negative, while measure **B** seems to improve for more positive values of **sd**. No clear pattern is observable in measure **C** as **sd** is varied, and measure **G** appears virtually unchanged for the values of **sd** used here. In broad terms, the trends seems similar to those suggested by table 4.1.1 for the ABI dataset (see previous section).

OVERALL	sd=0.1	sd=0.0	sd=-0.07	sd=-0.1
G	0.017578	0.017579	0.017579	0.017579
A	0.789313	0.770992	0.751145	0.745038
B	0.497396	0.536458	0.559896	0.575521
C	0.681424	0.684312	0.680462	0.682387

Table 4.2.1: Overall measures of error localisation performance for EPE data

The tables below show the performance for each value of **sd** with each of the variables in turn TOTINVTO and TOTEXPT. These variables are of primary interest as described in the Euredit evaluation handbook.

TOTINVTO	sd=0.1	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.5	0.5	0.5	0.5
beta	0.159018	0.170758	0.173959	0.180363
delta	0.16333	0.174921	0.178082	0.184405
RAE	0.341322	0.341322	0.341322	0.341322
RRASE	0.163342	0.163342	0.163342	0.163342
RER	1393.857	1393.857	1393.857	1393.857
tj	1.701787	1.701787	1.701787	1.701787
AREm1	0.079452	0.075377	0.069128	0.066895
AREm2	9.381622	9.568206	9.572234	9.598862
MSE	1657.294	1657.294	1657.294	1657.294

Table 4.2.2: Error localisation results for EPE variable TOTINVTO

TOTEXPTO	SD=0.1	sd=0.0	sd=-0.07	sd=-0.1
alpha	0.571429	0.571429	0.571429	0.5
beta	0.162353	0.171765	0.172941	0.177647
delta	0.168981	0.178241	0.179398	0.18287
RAE	-0.00114	-0.00114	-0.00114	-0.00104
RRASE	0.000366	0.000366	0.000366	0.000366
RER	2.345455	2.345455	2.345455	2.345455
tj	-0.00629	-0.00629	-0.00629	-0.00576
AREm1	0.044353	0.053333	0.063145	0.065808
AREm2	0.041279	0.043044	0.043998	0.04056
MSE	36.42854	36.42854	36.42854	34.55382

Table 4.2.2: Error localisation results for EPE variable TOTEXPTO

5 Imputation results

The procedure used during imputation experiments involved first creating a specification file for each dataset, and creating five copies of this file. Each copy was edited so that a different imputation mode (from NN, RANDOM, MEAN, WEIGHTED-MEAN, and MEDIAN) was selected in each case (for the SARs dataset only two imputation modes were investigated due to lack of time). A simple DOS batch file was used to repeat the imputation experiment five times each time using a different imputation mode (as indicated in the corresponding specification file). In most cases the system was left to choose default values for k (i.e. the k-NN) and the other parameters. An exception to this was the SARs dataset where the value of k was specified as 50 since the system default would have been larger than necessary in this case.

With the exception of section 5.2 (see that section for details), all the tables in section 5 are arranged in the same format with five results columns, one for each imputation mode as described earlier. Shading is used to highlight the better-scoring values in each row, with the darker shading used for the best scoring value(s) and lighter shading used for the second best scoring value(s). The table below provides a brief guide to the measured criteria for continuous variables used in each table.

Criteria measured	Reference	Additional comments
Slope		See page 19 in (Chambers, 2000). Huber M estimate used. Ideally should be close to 1.
t-val		Ideally close to 0. Measures how far Slope is from 1.
mse		Mean squared error, ideally 0.
R^2		Squared correlation coefficient, ideally close to 1.
dL1	equation 19	
dL2	equation 20	
dLinf	equation 21	
K-S	equation 25	Kolmogorov-smirnov
K-S_1	equation 26	alpha = 1
K-S_2	equation 26	alpha = 2
m_1	equation 28	k = 1
m_2	equation 28	k =2
MSE	equation 30	

Table: Measured criteria for continuous variables

In section 5.2 some tables use criteria for categorical variables instead, as shown below:

Criteria measured	Reference	Additional comments
W	equation 14	Measures preservation of marginal distribution
D	equation 15	Measures preservation of true values.
Eps	equation 16	

Table: Measured criteria for categorical variables

Equation numbers used in the tables above refer to those described in (Chambers, 2000).

5.1 Danish Population Register/Labour Force Survey LFS

The dataset consists of Danish population register records for individuals selected for interview for the Danish Labour Force Survey (1996). The dataset is adapted for use in the Euredit project to test imputation for the income variable by treating this as missing for those individuals who did not respond to the Labour Force Survey. The results reported in this section are based on the Y2 version of the dataset which contains missing values but not errors.

INCOME	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.831936	0.77263	0.886095	0.886242	0.946284
t-val	-35.190195	-41.413596	-28.807669	-28.77147	-13.886186
mse	9569009086	1.19E+10	6652994619	6648029533	6679189338
R^2	0.262119	0.144952	0.423972	0.424417	0.423124
dL1	62390.94419	74261.18252	48783.28862	48754.33796	45131.97126
dL2	102680.036	117357.4575	81746.92193	81713.57872	81156.99662
dLinf	850380	884379	833999	834028	834249
K-S	0.05988	0.08479	0.148982	0.149222	0.13485
K-S_1	0.010871	0.013568	0.02611	0.026089	0.021769
K-S_2	0.000356	0.000659	0.001908	0.001903	0.001105
m_1	7290.365988	10728.29533	11515.65509	11498.89461	1131.812216
m_2	888155313.9	2544173608	3047531467	3050703655	6904997159
MSE	4904609.742	9367060.272	10488350.59	10460707.21	1047917.18

Table 5.1.1: Imputation results for LFS variable INCOME

5.2 UK Sample of Anonymised Records (SARs)

This is a sample from a Census dataset. The data selected were a 1% sample of household records which provides the largest dataset in the collection. The data set contains information on people within households and therefore has a hierarchical structure. From Census documentation, patterns of errors and missingness in the pre-edited data can be recreated in the data distributed to participants. The results reported in this section are based on the Y2 version of the dataset which contains missing values but not errors.

The tables in section 5.2 are arranged in a slightly different format with two variables presented in each table and two results columns under each variable, one for each of two imputation modes (WEIGHTED MEAN and MEDIAN). Shading is used to highlight the better-scoring values under each variable, in each row.

	AGE		HOURS	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
AREm1	606.990123	306.987492	574.317757	294.424115
AREm2	606.990123	306.987492	574.317757	294.424115
Slope	0.740835	0.752259	1.125586	0.996535
t-val	-7.223066	-12.023619	1.862356	-0.187487
mse	98.99608	79.919944	126.452441	210.257492
R^2	0.417793	0.454998	0.000212	0.081173
dL1	9.370968	8.075	9	8.115385
dL2	12.606322	11.466255	11.677971	14.359398
dLinf	40	46	26	49
K-S	0.403226	0.45	0.5	0.230769
K-S_1	0.104839	0.082976	0.157095	0.037019
K-S_2	0.025982	0.021176	0.042335	0.00282
m_1	6.66129	5.991667	6.375	3.115385
m_2	313.725806	273.441667	404.75	208.346154
MSE	0.001056	0.001058	0.001181	0.001181

Table 5.2.1: Imputation results for SARs variables AGE and HOURS

	SEX		RELAT	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
W	0.692308	0.72	31.55965	35.439104
D	0.448276	0.416667	0.796296	0.576923
Eps	0.172414	0.277223	0.673458	0.449361

Table 5.2.3: Imputation results for SARs variables SEX and RELAT

	MSTATUS		LTILL	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
W	18.379642	21.265306	0.333333	2.666667
D	0.52459	0.2625	0.048387	0.053097
Eps	0.348027	0.126715	0	0

Table 5.2.3: Imputation results for SARs variables MSTATUS and LTILL

	TENURE		HHSPTYPE	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
W	28.409688	9.806161	4.266492	7.730595
D	0.868421	0.317073	0.5	0.398305
Eps	0.750733	0.134553	0.314305	0.255489

Table 5.2.4: Imputation results for SARs variables TENURE and HHSPTYPE

	ROOMSNUM		BATH	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
W	25.883317	52.357946	0	0
D	0.557143	0.618321	0	0
Eps	0.398064	0.510365	0	0

Table 5.2.5: Imputation results for SARs variables ROOMSNUM and BATH

	CENHEAT		INSIDEWC		CARS	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
W	28.2	42	0	0	20.666667	26.287049
D	0.390244	0.346405	0	0	0.430769	0.413043
Eps	0.217779	0.215686	0	0	0.243607	0.282609

Table 5.2.6: Imputation results for SARs variables CENHEAT, INSIDEWC, and CARS

	COBIRTH		ISCO1		ISCO2	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
W	9.22006	18.181818	6.242737	53.879741	7.618073	43.091494
D	0.666667	0.157534	0.83871	0.852273	1	0.931034
Eps	0.524533	0.005609	0.694447	0.770328	1	0.862069

Table 5.2.7: Imputation results for SARs variables COBIRTH, ISCO1, and ISCO2

	DISTWORK		WORKPLACE		QUALNUM	
	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN	WTD MEAN	MEDIAN
W	1.750637	9.831418	6	4	16	36.222222
D	0.5	0.571429	0.888889	0.125	0.5	0.384615
Eps	0	0.323993	0.666667	0	0.25	0.230769

Table 5.2.8: Imputation results for SARs variables DISTWORK, WORKPLACE, and QUALNUM

5.3 UK Annual Business Inquiry (Retail Section)

The ABI dataset consists of a sample from the retail section of the U.K. Annual Business Inquiry (1999). For each business the dataset contains information on aspects of purchase costs and employment costs. The data is based on responses for either of 2 possible questionnaires. The larger questionnaire contains 17 retail questions. The smaller one only asks for summary information and contains 5 retail questions. Businesses on the dataset therefore either have 5 or 17 responses to individual questions. The results reported in this section are based on the Y2 version of the dataset, which contains missing values but not errors.

TURNOVER	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.601782	0.772398	1.026589	1.050274	1.210826
t-val	-14727.56982	-82.861624	15.537449	26.961219	140.775111
mse	421920267.1	3.06E+10	1.79E+10	16928770000	2.38E+10
R^2	0.988247	0.143547	0.815609	0.833653	0.639476
dL1	557.93525	878.917817	576.394886	570.580788	654.853699
dL2	20848.90588	29706.4243	23020.3066	22622.23535	27216.45232
dLinf	34611.58913	46667.4444	38124.16509	37451.4551	45054.26939
K-S	0.066176	0.058824	0.066176	0.066176	0.073529
K-S_1	0.004479	0.001965	0.001723	0.00181	0.001351
K-S_2	0.000036	0.000016	0.000023	0.000023	0.000017
m_1	303.514852	410.743686	391.621615	385.911458	559.172663
m_2	1722668335	875997246.3	919386274.3	912797936.9	989200177.2
MSE	1761889.389	1726690.292	1726094.029	1726181.195	1621242.672

Table 5.3.1: Imputation results for ABI variable TURNOVER

EMPTOTC	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	1.09085	0.928443	0.792165	0.796168	1.121222
t-val	17.876876	-81.330442	-365.759311	-344.529238	148.625277
mse	635276.4508	246500.8893	158029.5282	130879.8377	144650.1938
R^2	0.935381	0.942393	0.965425	0.970685	0.968387
dL1	23.078173	24.153703	18.630044	17.923052	16.593864
dL2	148.6216	85.982371	128.139593	118.924198	65.243937
dLinf	213.243839	59.232831	127.38729	124.422825	57.284754
K-S	0.067227	0.117647	0.142857	0.142857	0.109244
K-S_1	0.004453	0.007619	0.006715	0.006418	0.00669
K-S_2	0.000055	0.000082	0.000097	0.000091	0.000092
m_1	8.989751	5.808313	0.132834	0.870815	10.279398
m_2	73663.19221	8689.384494	83609.07462	77522.82136	18292.09693
MSE	11515.18438	11769.5952	13385.1099	13711.37853	11912.92138

Table 5.3.2: Imputation results for ABI variable EMPTOTC

PURTOT	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.973032	0.972862	1.210687	1.214306	1.324886
t-val	-151.747446	-167.691817	111.119707	109.857861	129.05372
mse	89457551.8	926868599.8	590017247	578118010.2	925192481.6
R^2	0.981438	0.727809	0.880817	0.88346	0.879428
dL1	107.723499	160.975172	157.504277	156.876758	183.837574
dL2	1367.576497	4737.118934	4489.373258	4466.439977	5606.733523
dLinf	1924.924198	7385.018571	6437.260945	6400.630941	7761.563018
K-S	0.071429	0.085714	0.078571	0.071429	0.078571
K-S_1	0.005038	0.002299	0.003846	0.003825	0.00433
K-S_2	0.000038	0.000022	0.00003	0.000029	0.000033
m_1	21.702495	93.081025	88.276592	87.836501	145.2628
m_2	13289115.75	42080503.12	50411609.38	50280970.81	60533989.37
MSE	2086083.859	2218505.717	2416710.508	2416715.814	2416305.685

Table 5.3.3: Imputation results for ABI variable PURTOT

TAXTOT	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.635972	0.245462	0.786586	0.80767	1.256017
t-val	-61.585718	-190.845269	-69.555706	-60.040456	67.918816
mse	233624.3473	275661.0602	32396.75789	29508.00728	17047.96696
R^2	0.3134	0.025357	0.907013	0.913852	0.939555
dL1	6.605313	9.273154	4.709212	4.610275	4.053619
dL2	79.535662	93.527176	28.511028	27.281857	28.219452
dLinf	122.812875	128.139627	24.312605	24.282846	34.311424
K-S	0.110236	0.102362	0.15748	0.15748	0.055118
K-S_1	0.004239	0.006675	0.004992	0.005151	0.005771
K-S_2	0.000079	0.000168	0.000067	0.000064	0.00006
m_1	0.949261	0.61096	1.211339	1.067939	1.651507
m_2	7643.780106	6721.69991	373.125456	246.875298	3244.615677
MSE	146.734747	151.191593	169.648034	163.781779	138.084805

Table 5.3.4: Imputation results for ABI variable TAXTOT

ASSACQ	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.610078	0.350497	2.299377	2.348604	11.8652
t-val	-10615.84579	-1176.787082	30979.85687	30741.27331	20877.74702
mse	1574981196	2318876295	37611487.75	34674621.05	1128629344
R^2	0.583305	0.02655	0.983912	0.985178	0.757272
dL1	122.991111	164.940762	103.40781	104.380197	153.759791
dL2	6022.316675	8174.93057	4750.88764	4817.449093	7985.846908
dLinf	11684.94758	15804.15437	8955.59911	9100.061642	15452.39291
K-S	0.325	0.24	0.36	0.36	0.21
K-S_1	0.001334	0.001221	0.001434	0.001396	0.001327
K-S_2	0.000008	0.000009	0.000009	0.000008	0.000012
m_1	83.915096	148.691998	97.402055	98.321647	152.408623
m_2	55789183.95	67380530.76	55118467.42	55619013.91	67375916.82
MSE	5199.377939	7685.053967			1729.589657

Table 5.3.5: Imputation results for ABI variable ASSACQ

ASSDISP	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.052253	0.000001	0.096045	0.072863	
t-val	-3933.82518	-48821986.2	-1069.85705	-1875.963002	
mse	814498.3729	890484.1787	768226.1541	796539.67	
R^2	0.909069	0.214256	0.978925	0.978319	
dL1	7.783774	8.308057	6.065148	6.112376	5.410936
dL2	48.95104	141.884922	44.113169	43.821307	150.748872
dLinf	61.11398	239.017337	68.991516	68.206566	257.379565
K-S	0.355263	0.368421	0.644737	0.644737	0.203947
K-S_1	0.004985	0.001537	0.005846	0.005911	0.001763
K-S_2	0.000045	0.000016	0.000155	0.000159	0.000059
m_1	3.447436	6.163478	3.329501	3.322842	5.319427
m_2	5706.417211	22542.37296	10864.34337	10747.61477	24740.63834
MSE	151.400755	166.628831	430.378506	411.088259	101.873028

Table 5.3.6: Imputation results for ABI variable ASSDISP

5.4 Swiss Environment Protection Expenditures (EPE)

This is a Business Survey containing categorical variables and a large number of true zero responses (i.e. where there was no expenditure) and also contains outliers. The originators of the data themselves recreated the pattern of errors and missingness, and edit rules were also supplied. The results reported in this section are based on the Y2 version of the dataset, which contains missing values but not errors.

TOTINVTO	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.611244	0.380364	0.797776		4.186055
t-val	-17.326554	-24.655258	-6.514134	-6.543223	21.070571
mse	970378.7171	1150254.651	979653.1805	979666.4041	938406.5043
R^2	0.172545	0.013176	0.155711	0.156337	0.18122
dL1	104.944681	118.468714	104.430676	104.112756	105.125946
dL2	278.196426	331.088521	281.291029	281.280737	300.096485
dLinf	315.751565	349.767152	324.19278	324.19278	346.256944
K-S	0.477778	0.666667	0.388889	0.388889	0.622222
K-S_1	0.048276	0.048955	0.049656	0.049622	0.061365
K-S_2	0.007766	0.011523	0.007185	0.007189	0.014916
m_1	61.825594	88.586389	57.121631	57.771163	104.724082
m_2	77072.02138	69295.54836	86143.03344	86208.83018	95873.53029

Table 5.4.1: Imputation results for EPE variable TOTINVTO

TOTEXPTO	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.551205	0.14473	0.797214	0.793996	3.028339
t-val	-53.625532	-99.51939	-25.353985	-25.111735	42.715912
mse	828826.3701	951715.0934	792351.9331	792929.0557	782248.9019
R^2	0.141578	0.010692	0.179699	0.179261	0.219717
dL1	67.696284	89.380771	52.743278	52.984982	68.396786
dL2	279.353386	394.325584	271.30789	271.376386	289.983297
dLinf	548.92523	556.915131	551.493412	551.493412	563.573382
K-S	0.4	0.554286	0.308571	0.314286	0.537143
K-S_1	0.011052	0.011612	0.010107	0.010199	0.017986
K-S_2	0.00146	0.002679	0.001056	0.001072	0.003942
m_1	24.879961	33.804602	31.962372	32.1643	68.344725
m_2	68697.49446	6187.716868	77339.89128	77307.95825	90521.79572

Table 5.4.2: Imputation results for EPE variable TOTEXPTO

5.5 European Community Household Survey (GSOEP)

This is the only social survey dataset in the collection suitable for edit purposes and also the only social data with a longitudinal aspect. It consists of information from a panel of people interviewed over a number of years. There is also an element of hierarchical data with information on people within households. There is documentation on the full dataset allowing accurate proportions of errors and missing values to be generated. The results reported in this section are based on the Y2 version of the dataset, which contains missing values but not errors. Results are shown here only for the first three years of the survey (years 1991 to 1993) to provide an indication of the form that the results take. Full results will be taken into account during WP6.

income 91	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.902863	0.802044	0.96822	0.970108	0.976106
t-val	-15.793043	-24.708059	-5.460485	-5.384541	-4.545728
mse	384118686.5	586226310.1	225058111.9	219919920.5	259818820.9
R^2	0.540427	0.334582	0.70222	0.708916	0.655495
dL1	11273.97092	14394.95727	9444.636202	9328.405935	9042.414243
dL2	20434.52508	26869.05781	14985.95958	14817.2745	16086.52403
dLinf	339996	512500	172994	172272	179400
K-S	0.033234	0.063501	0.278932	0.278932	0.065282
K-S_1	0.002323	0.004337	0.016315	0.016415	0.014013
K-S_2	0.000041	0.000167	0.001032	0.001025	0.00053
m_1	31.537685	2116.655786	997.632047	1003.039763	572.862908
m_2	70826506.32	299389860.1	197631767.3	191756480.4	269230490.8
MSE	216916.0517	666539.224	310609.5545	312135.517	224910.7036

Table 5.5.1: Imputation results for GSOEP variable INCOME 91

hhinco 91	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.858665	0.829254	0.993704	0.991347	1.084167
t-val	-13.435136	-14.120848	-0.656378	-0.90189	7.962435
mse	1830518862	2154849540	1076758373	1072332257	1058187956
R^2	0.215367	0.124011	0.388821	0.391336	0.405533
dL1	28732.87953	32259.57329	23192.35905	23114.00119	21908.37092
dL2	44952.16602	48813.32789	32785.14123	32711.96597	33161.16482
dLinf	518500	582267	211511	206670	240482
K-S	0.052819	0.068843	0.140059	0.140653	0.2
K-S_1	0.005376	0.006276	0.049866	0.049187	0.049207
K-S_2	0.000113	0.000198	0.004524	0.004371	0.005479
m_1	1778.127003	3356.273591	1393.826113	1297.395846	7486.046884
m_2	47350093.77	223261085.9	1250348127	1224890812	1864626982
MSE	826400.7017	1618025.512	620567.0413	596376.7678	5905158.496

Table 5.5.2: Imputation results for GSOEP variable HHINCO 91

income 92	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.923717	0.843288	1.025601	1.025984	1.035041
t-val	-13.124856	-15.617093	4.284043	4.493228	6.887363
mse	476276712.3	903333208.4	278243595.2	275557022.8	283814505
R^2	0.493903	0.215259	0.679854	0.683277	0.675976
dL1	12392.19412	17973.52131	9666.40553	9572.783986	9220.528226
dL2	21942.0722	30839.98033	16768.36578	16695.81001	16979.85021
dLinf	192000	385420	184906	185704	188050
K-S	0.08871	0.222926	0.194124	0.195276	0.077765
K-S_1	0.012889	0.018886	0.018408	0.018556	0.015604
K-S_2	0.00045	0.002388	0.000818	0.000829	0.000603
m_1	3138.384217	8494.228687	2027.048387	2079.340438	2930.411866
m_2	313387030.8	468877940.2	431458276.5	437424418	460637572.2
MSE	1235842.611	7726429.353	639605.0762	661783.9446	1091184.433

Table 5.5.3: Imputation results for GSOEP variable INCOME 92

hhinco 92	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.873062	0.807108	1.053956	1.051828	1.1395
t-val	-10.633327	-11.468709	4.706092	4.451756	11.490774
mse	2368580048	3387923349	1588912574	1589161355	1678927163
R^2	0.182416	0.069087	0.304633	0.304978	0.304032
dL1	33103.61982	42286.22062	27025.32085	27063.28226	26777
dL2	49369.869	59173.96342	40173.04845	40153.37411	42114.3568
dLinf	449598	534239	495297	496043	503099
K-S	0.112327	0.288018	0.152074	0.147465	0.207373
K-S_1	0.014579	0.03295	0.018514	0.018259	0.024282
K-S_2	0.000939	0.006046	0.001686	0.001632	0.002916
m_1	8926.75576	19857.77362	8144.154954	8025.436636	14757.65438
m_2	1311776310	2203965248	2284463896	2260967937	2871310801
MSE	8802426.76	41555953.02	7348430.811	7149790.444	23089703.6

Table 5.5.4: Imputation results for GSOEP variable HHINCO 92

income 93	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.932205	0.846262	1.078636	1.077577	1.072109
t-val	-9.527275	-11.946876	11.390366	11.524786	11.924878
mse	756445111.9	1160132622	405791573.5	396192822.5	521993586.4
R^2	0.3365	0.175106	0.601021	0.610501	0.50696
dL1	14452.45286	21199.44939	10750.01099	10731.71718	11194.25275
dL2	27611.76319	34297.34355	20585.59116	20348.50343	23118.09675
dLinf	479998	408000	337860	303290	412166
K-S	0.139965	0.371891	0.144592	0.144592	0.114517
K-S_1	0.011587	0.028918	0.011674	0.011469	0.013105
K-S_2	0.000815	0.006717	0.000808	0.000776	0.000826
m_1	5792.036437	14098.15732	4642.119722	4583.377097	6521.898785
m_2	586105858.9	934650490.1	738331080.3	729884741.4	784131275.2
MSE	3710993.794	20761214.52	2467241.157	2411823.059	4623595.947

Table 5.5.5: Imputation results for GSOEP variable INCOME 93

hhinco 93	NN	RANDOM	MEAN	WTD MEAN	MEDIAN
Slope	0.888293	0.778109	1.153474	1.150964	1.224814
t-val	-8.623481	-9.092754	11.644653	11.333498	15.200934
mse	3384484854	5184645349	2214800812	2207922336	2641590418
R^2	0.110915	0.029175	0.250267	0.252956	0.198705
dL1	36903.89647	49975.00694	29657.32678	29610.19896	31334.02198
dL2	58539.19247	74247.61915	48335.80679	48218.24861	53140.95348
dLinf	790999	791000	732894	732226	736640
K-S	0.176403	0.46096	0.181029	0.176981	0.235975
K-S_1	0.019144	0.041636	0.021172	0.020932	0.028278
K-S_2	0.001974	0.012764	0.002333	0.002266	0.003994
m_1	15509.3181	32984.90168	14707.23193	14557.79526	22766.68016
m_2	2580008707	3282349571	3598809116	3568181122	4186729063
MSE	25413583.76	112939711.9	22848066.29	22398107.66	54003915.07

Table 5.5.6: Imputation results for GSOEP variable HHINCO 93

6 Discussion

It is not possible to derive many general conclusions about the results presented here because they are based on the use of a single basic approach – CMM. Later during WP6 of the Euredit project these CMM results will be compared with the results obtained in other work packages using different methods, and then a clearer picture of performance in relative terms should be become available.

The results presented in sections 4 and 5 suggest that the York CMM-based system is at least partially successful in identifying errors and it generally appears an effective tool for imputation, especially where a large degree of automation is required.

For error localisation, it appears that a wider variation in the parameter **sd** should have been investigated. Part of the reason for the limited variation is due to the evaluation process, which requires that results be returned to another partner (ONS) for evaluation against the (withheld from other partners) true data. Although this is a scientifically sound approach, it places a practical limit on the number of experimental results that can be evaluated in a given time. Nevertheless, it is apparent that there is some trade-off between the A and B scores, and between the alpha and beta scores, in most cases. This suggests that the cost of improving the error detection rate is an increase in the false-alarm rate.

For imputation, the WEIGHTED-MEAN mode seems to offer comparatively good performance in most dataset/variable combinations for many of the evaluation criteria, but not in every case. MEDIAN-MODE also appears to good (but not always the best) performance in a range of situations. It is very difficult to draw general conclusions about choice of best imputation mode on the basis of these results without a deeper more careful analysis. However, it does seem that it may be possible to identify a policy for using a particular imputation mode according to data type and the criteria of greatest importance in a given situation. For example, the NEAREST-NEIGHBOUR mode often performs good-to-best for preservation of distribution according to the Kolmogorov-Smirnov criteria, whereas the WEIGHTED-MEAN mode often performs good-to-best for preservation of values (criteria dL1, dL2, and dLinf).

Overall, the results so far seem encouraging but it is not possible to judge performance in relative terms at this stage since results for other systems are not yet available for comparison. Also, more experiments need to be performed to better characterise the system. If time permits, the current system will be extended with the aim of identifying errors in particular values rather than errors at record level.

References

Austin, J., (1996) AURA, A distributed associative memory for high-speed symbolic reasoning. In: Sun, R. (ed.) **Connectionist Symbolic Integration**. Kluwer.

Austin, J. and Kennedy, J. (1998) PRESENCE, a hardware implementation of binary neural networks. In: Perspectives in Neural Computing, ICANN98, Skövde, Sweden, 1998. Springer Verlag.

Austin, J. and O'Keefe, S., (1999) Neural Networks for the Estimation of Missing Data, Euredit Report T009.01.

Byers, S. and Raftery A.E., (1996) Nearest neighbour clutter removal for estimating features in spatial point processes, TR 305, Department of Statistics, University of Washington, Seattle.

Chambers, R, (2000) Evaluation Criteria for Statistical Editing and Imputation, available as report no. 28 in the National Statistics Methodology Series, and also as Euredit deliverable D3.3.

Fisher, R.A., (1936) The use of multiple measurements in taxonomic problems, Annual Eugenics, 7 (1936), 179-188.

Hodge, V. et al (2002) *currently untitled paper in preparation – aims to compare use of CMM to find k nearest neighbours with conventional methods.*

Steinbuch, K. (1961), Die lernmatrix, Kybernetic, 1, pp. 36-45.

Turner, M. and Austin, J. (1997) Matching performance of binary correlation matrix memories, Neural Networks 10, 1637-1648.

Willshaw, D.J., Buneman, O.P., and Longuet-Higgins, (1969) H.C., Non-Holographic Associative Memory, Nature 222 (1969) 960-962.

Zhou, P., Austin, J. and Kennedy, J. (1999) A high performance K-NN classifier using a binary correlation matrix memory. Advances in Neural Information Processing Systems Vol. 11.

Appendix A: Review of the KNN method

The conventional Hot Deck approach – see for example (Kalton, 1983) – may be defined as a method where an imputed value is selected from an empirical distribution, consisting of values from similar responding units. The units which comprise the empirical distribution (sometimes referred to as donor records) are selected based on some similarity metric, and there are a number of methods by which the selection from the distribution is made. The most straightforward implementations of the Hot Deck approach construct the distribution by selecting records which match exactly on the variables which are present, and then select some value from this distribution. The number of matching records may be small.

The K-NN algorithm is closely related to the conventional Hot Deck approach. The difference between the techniques lies in the way in which the donor sets are derived. Whereas in the basic Hot deck technique the donor records match exactly on the (subset of) available data items, the KNN technique requires a distance metric to be defined so that the closest matching exemplars from the training data can be selected. These are used as the basis for computation of the required output for a missing value.

The K-NN method is a widely used and simple method for data classification. The following argument justifies the KNN technique in terms of the construction of a Bayesian classifier. We posit that the training data represents the output density function such that the probability that a new case x drawn from the unknown density function p(x) will fall inside some region R of the output space or x-space is by definition:

$$P = \int_R p(x')dx'$$

If we have N data points drawn independently from p(x) then the probability that K of them will fall in the region R is given by the binomial distribution:

$$\Pr(K) = \frac{N!}{K!(N-K)!} P^K (1-P)^{N-K}$$

The expected fraction of the N points in this region R is $E[K/N] = P$, and the variance is:

$$E = \left[\left(\frac{K}{N} - P \right)^2 \right] = \frac{P(1-P)}{N}$$

Thus as N tends to ∞ the variance tends to zero, and in this limit we obtain the actual probability distribution for the output space. The approximation $P = K/N$ is unbiased.

Assuming p(x) is continuous and doesn't vary over the region R then we can approximate the integral for P by:

$$P = \int_R p(x')dx' \approx p(x)V$$

where V is the volume of the region R and x lies inside R. From these equations we obtain the intuitive results, that

$$p(x) \approx \frac{K}{NV}$$

Note that two assumptions are required to make this approximation. R is assumed to be large so that $P = K/N$ is valid, and R is required to be small to give $P = p(x)V$. A compromise is required to obtain the best approximation for $p(x)$.

In practice, we can choose for V to be fixed (which gives the Kernel-based density estimation methods), or K to be fixed to give the K-nearest neighbour (KNN) techniques. Duda and Hart (Duda and Hart, 1973) showed that the KNN technique does in fact converge to the true density function.

Where the distribution of the data-points in output space is non-uniform, the kernel-based (fixed V) techniques can over-smooth is some regions and under-smooth in others. This is avoided by using the KNN techniques.

A Bayesian classifier may be constructed thus. Suppose that the data contains N_k points in each of k classes Ck, such that

$$\sum_k N_k = N$$

Take a hypersphere around the test point x that encloses K training points in a volume V. Then the class-conditional densities are estimated by

$$p(x \mid C_k) = \frac{K_k}{N_k V}$$

The unconditional density is still estimated by

$$p(x) \approx \frac{K}{NV}$$

The prior probabilities are estimated by

$$P(C_k) = \frac{N_k}{N}$$

Bayes' theorem then gives

$$P(C_k \mid x) = \frac{p(x \mid C_k)P(C_k)}{p(x)}$$

Thus the largest estimate for K_k indicates the class C_k to which is the most probable class for the test vector x. The use of this technique requires a metric to be defined which allows the distances between test and training vectors to be calculated.

The description given above is of the Bayesian classifier constructed using the k-NN technique. If, instead of a classification problem, we have a function estimation problem there are two routes which can be taken. We can either reduce the continuous values of the function to categorical data, by some quantisation algorithm,

or we can calculate the required value as a function of the values of the K nearest neighbours. We have taken the latter approach.

For more details of this technique and the construction of classifiers, see (Bishop, 1995).

A comparison between K-NN methods and other machine learning and statistical methods has been given in Michie (1994). This showed that the K-NN method is always as good if not better than any other methods across many different problems.

References

Bishop, C. M. (1995) Neural Networks for Pattern Recognition, Oxford University Press, Oxford.

Duda, R. O. and Hart, P. E. (1973) Pattern Classification and Scene Analysis. John Wiley & Co, New York.

Kalton, G. (1983) Compensating for missing survey data, Survey Research Center, Institute for Social research, University of Michigan.

Michie, D., Spiegelhalter, D. J., Taylor, C. C., (1994) Machine Learning, Neural and Statistical Classification, Prentice Hall.

Appendix B: Some experiments with the Iris database

The effectiveness of the DKN strategy has been investigated using a simple dataset based on the well-known "Iris Plant database (Fisher, 1936). The original iris database contains 150 records representing 3 classes described in terms of 4 numeric attributes. It becomes difficult to visualise the relative positions of data in higher dimensions so a much simpler 2 attribute dataset was derived from the original using a subset based only on the first 2 attributes of each record. It was necessary to remove duplicate records from the subset to avoid multiple overlaid data points, and the data was sorted on the first attribute (highest to lowest) and then on the second attribute (highest to lowest) resulting in a highly-ordered dataset which is easily viewed as a 2D graph. The modified Iris database is shown in Appendix C.

Here the **DKN** strategy was applied to the results of CMM matching (as described in section 2.5) to produce a re-ordered list of the original data, ranked in order of distance from each point to its k^{th} neighbour. Values of k were selected between 3 to 30.

The pre-processing method used in these experiments is the original method (see sections 2.4.1 to 2.4.5) in which adjacent bits are set in the binary input vectors representing continuous values, rather than the more recent method which uses weighted binary input vectors. The difference in accuracy is more apparent for larger values of k when finding the k nearest neighbours. Because the experiments with the iris data use only small values for k, the results are still meaningful, and the overall operation of the DKN strategy is not significantly affected, though sometimes a few points may be ranked higher in the list of outlier candidates than they should be.

Figure B.1 opposite shows a 2D scatter plot of first 2 columns of the iris data used in this series of experiments. The scatter plot is constructed using the values of sepal-width and sepal-length (respectively) taken from the iris data as X-Y coordinates. These were used as input to the outlier identification procedure. It is not easy to identify clear clusters of data points in this 2D view of the data, but 2 main groups of points have been "ringed" in one possible interpretation of the data in terms of clusters. In this interpretation, the data points remaining outside the ringed clusters could be considered outliers. This is rather subjective, of course, as larger rings could easily be drawn to include more data points.

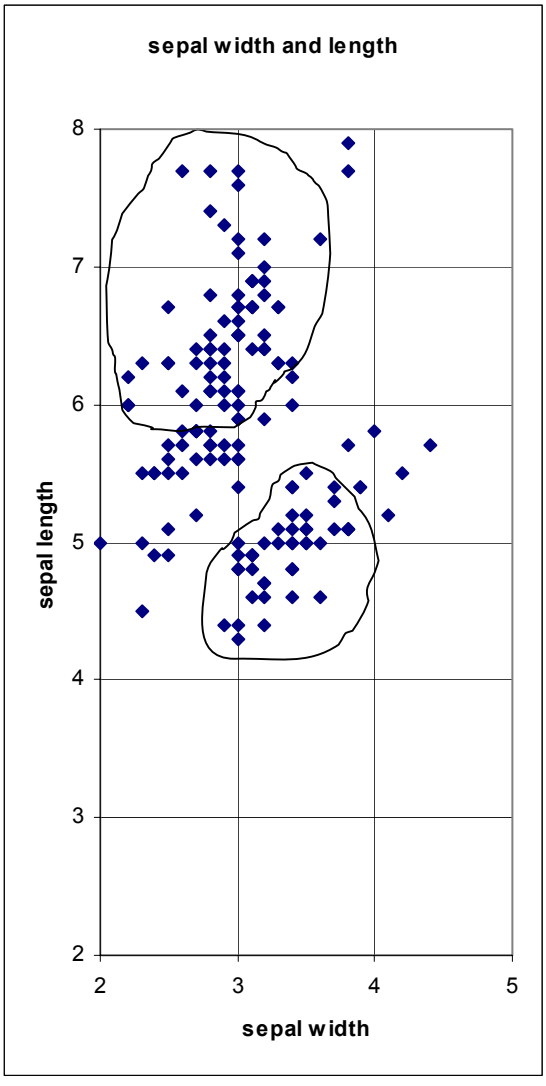


Figure B.1: Scatter graph of the first 2 columns for the iris database (sepal length and sepal width).

Figures B.2 (a) to (h) on the following pages summarise the results of the DKN strategy applied to the modified iris

data for a range of values of k from 3 to 30. The data plotted in this series of figures represents the top eleven data points identified by the DKN ranking strategy. Actual rankings are not shown in the graphs. For reference the top eleven ranked data points produced in each experiment are reproduced in Appendix D.

A number of interesting observations can be drawn from these results. In all cases, "visually" extreme points are ranked in the top eleven (i.e. identified as "outliers"). The changing positions of the top eleven ranked points suggests that the method is quite sensitive to the choice of k (as might be expected). Although the results appear graphically similar, the actual changes in rank order are vary considerably (see Appendix D). It remains unclear from these figures what choice of k-value might be considered "best" in the sense of correctly identifying outliers. Part of the problem is that there is no "ground truth" to indicate that certain points are somehow in error or different in a physical sense, because we have merely chosen some data convenient for plotting and visualising in 2D.

Distance to 30th neighbour	sepal-length	sepal-width	petal-length	petal-width	class
2.72074	5.7	4.4	1.5	0.4	Iris-setosa
2.57588	7.9	3.8	6.4	2	Iris-virginica
2.46891	7.7	3.8	6.7	2.2	Iris-virginica
2.44844	5.5	4.2	1.4	0.2	Iris-setosa
2.23971	5.8	4.0	1.2	0.2	Iris-setosa
2.23971	5.0	2.0	3.5	1	Iris-versicolor
2.21713	5.2	4.1	1.5	0.1	Iris-setosa
2.02011	5.4	3.9	1.7	0.4	Iris-setosa
1.97579	4.5	2.3	1.3	0.3	Iris-setosa
1.85835	7.7	2.6	6.9	2.3	Iris-virginica
1.85292	7.2	3.6	6.1	2.5	Iris-virginica

Table B.1: Top-eleven records produced by DKN ranking

For convenience, a typical example of the system output is given above showing the eleven top-ranked data points for k=30, and this data is plotted in Figure B.2 (h). The detailed results for this example are shown in Appendix D for all points (but clearly the points nearer the top of the list are of greatest interest). Additionally a threshold criterion is used in an attempt to make a firm decision about which points should be considered outliers. The DKN strategy assumes outliers have a relatively large distance to their k^{th} neighbour. The threshold here is determined using an approximation to the first derivative of the DKN function. Thus a threshold is selected by finding the position of largest change in ranked list of DKN distances, which is assumed to represent the difference in cluster sizes between relatively large *normal* data clusters and relatively small *outlier* clusters.

For the experiment described here, the data point in the ranked list which is ranked just above the position at which the largest change in DKN distance occurs is 2.23971, so we label this point and all data points ranked above in the list as likely outliers. The top four data points from table B.1 are thus selected as shown in table B.2 below.

Likely Outliers: These are assumed to have large distance to the k=30th neighbour.
The distance prior to which the largest change occurs is: 2.23971. 4 records are likely outliers.

2.72074	5.7	4.4	1.5	0.4	Iris-setosa
2.57588	7.9	3.8	6.4	2.0	Iris-virginica
2.46891	7.7	3.8	6.7	2.2	Iris-virginica
2.44844	5.5	4.2	1.4	0.2	Iris-setosa

Table B.2: Top-four records produced by threshold criterion

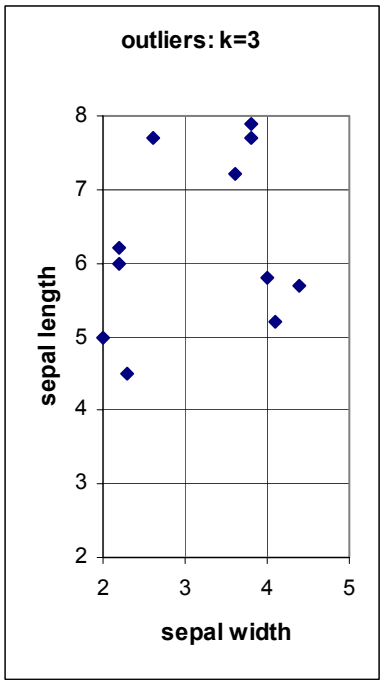


Figure B.2 (a)

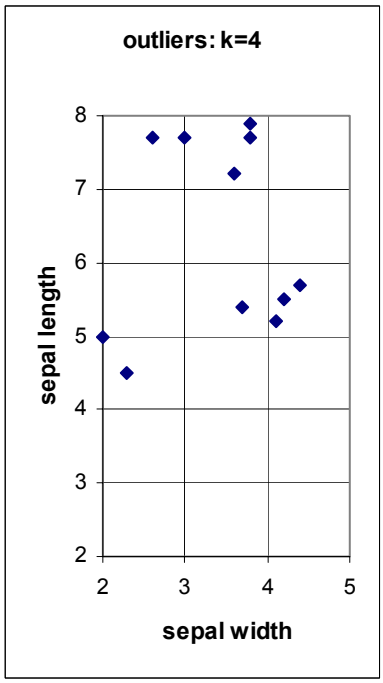


Figure B.2 (b)

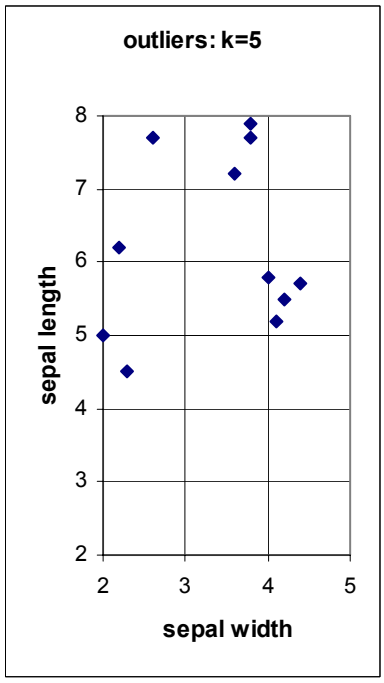


Figure B.2 (c)

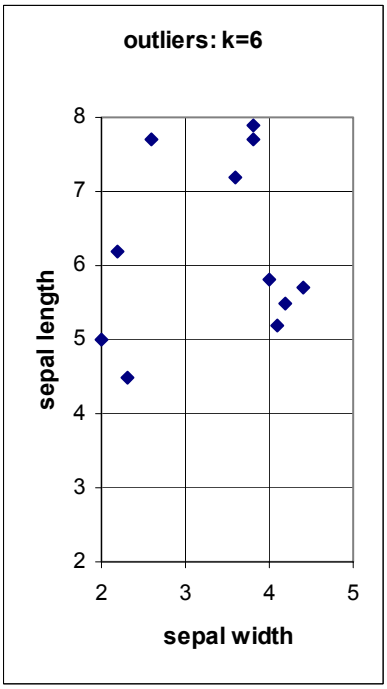


Figure B.2 (d)

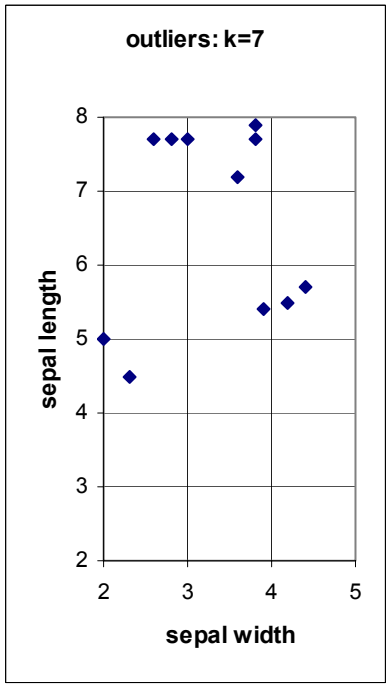


Figure B.2 (e)

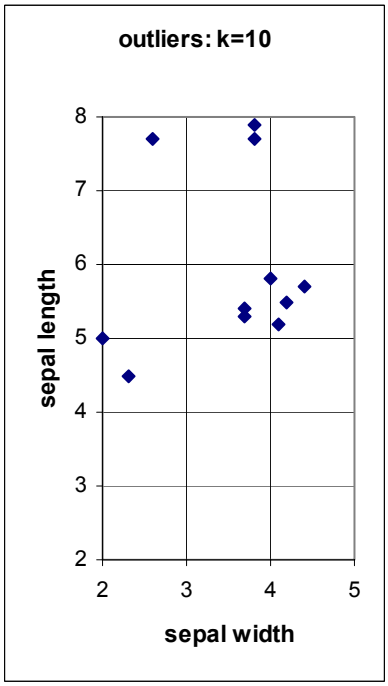


Figure B.2 (f)

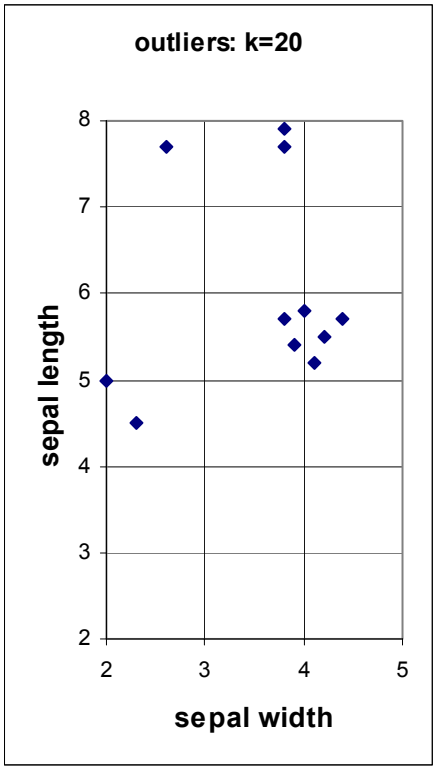


Figure B.2 (g)

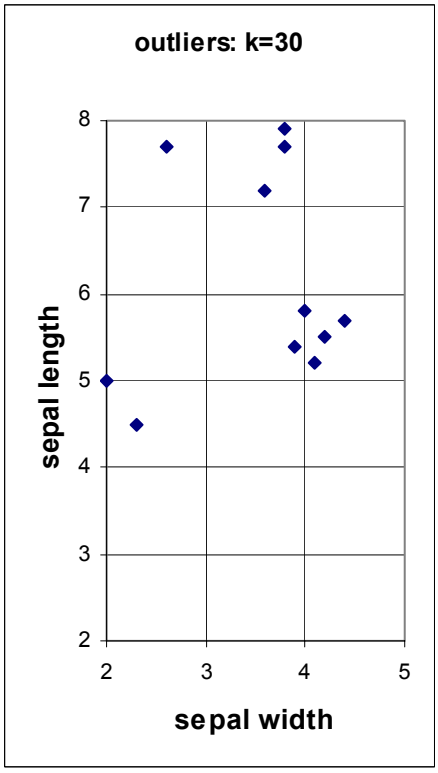


Figure B.2 (h)

References

Fisher, R.A., (1936) The use of multiple measurements in taxonomic problems, Annual Eugenics, 7 (1936), 179-188.

Appendix C: Details of the modified Iris database

The modified Iris database used in experiments is shown in Table C (a)-(d) below. Note that all attributes are shown, even though only the first two – sepal-length and sepal-width – are actually used.

sepal-length	sepal-width	petal-length	petal-width	class
7.9	3.8	6.4	2	Iris-virginica
7.7	3.8	6.7	2.2	Iris-virginica
7.7	3.0	3.5	1.0	Iris-versicolor
7.7	2.8	1.3	0.3	Iris-setosa
7.7	2.6	1.5	0.4	Iris-setosa
7.6	3.0	3.9	1.4	Iris-versicolor
7.4	2.8	6.1	2.3	Iris-virginica
7.3	2.9	6.7	2.0	Iris-virginica
7.2	3.6	6.9	2.3	Iris-virginica
7.2	3.2	6.6	2.1	Iris-virginica
7.2	3.0	6.1	1.9	Iris-virginica
7.1	3.0	6.3	1.8	Iris-virginica
7.0	3.2	6.1	2.5	Iris-virginica
6.9	3.2	6.0	1.8	Iris-virginica
6.9	3.1	5.8	1.6	Iris-virginica
6.8	3.2	5.9	2.1	Iris-virginica
6.8	3.0	4.7	1.4	Iris-versicolor
6.8	2.8	5.7	2.3	Iris-virginica
6.7	3.3	4.9	1.5	Iris-versicolor
6.7	3.1	5.9	2.3	Iris-virginica
6.7	3.0	5.5	2.1	Iris-virginica
6.7	2.5	4.8	1.4	Iris-versicolor
6.6	3.0	5.7	2.1	Iris-virginica
6.6	2.9	4.4	1.4	Iris-versicolor
6.5	3.2	5.0	1.7	Iris-versicolor
6.5	3.0	5.8	1.8	Iris-virginica
6.5	2.8	4.4	1.4	Iris-versicolor

Table C (a)

sepal-length	sepal-width	petal-length	petal-width	class
6.4	3.2	4.6	1.3	Iris-versicolor
6.4	3.1	5.1	2.0	Iris-virginica
6.4	2.9	5.8	2.2	Iris-virginica
6.4	2.8	4.6	1.5	Iris-versicolor
6.4	2.7	4.5	1.5	Iris-versicolor
6.3	3.4	5.5	1.8	Iris-virginica
6.3	3.3	4.3	1.3	Iris-versicolor
6.3	2.9	5.6	2.1	Iris-virginica
6.3	2.8	5.3	1.9	Iris-virginica
6.3	2.7	5.6	2.4	Iris-virginica
6.3	2.5	4.7	1.6	Iris-versicolor
6.3	2.3	5.6	1.8	Iris-virginica
6.2	3.4	5.1	1.5	Iris-virginica
6.2	2.9	4.9	1.8	Iris-virginica
6.2	2.8	4.9	1.5	Iris-versicolor
6.2	2.2	4.4	1.3	Iris-versicolor
6.1	3.0	5.4	2.3	Iris-virginica
6.1	2.9	4.3	1.3	Iris-versicolor
6.1	2.8	4.8	1.8	Iris-virginica
6.1	2.6	4.5	1.5	Iris-versicolor
6.0	3.4	4.6	1.4	Iris-versicolor
6.0	3.0	4.7	1.4	Iris-versicolor
6.0	2.9	4.0	1.3	Iris-versicolor
6.0	2.7	5.6	1.4	Iris-virginica
6.0	2.2	4.5	1.6	Iris-versicolor
5.9	3.2	4.8	1.8	Iris-virginica
5.9	3.0	4.5	1.5	Iris-versicolor

Table C (b)

sepal-length	sepal-width	petal-length	petal-width	class
5.8	4.0	5.1	1.6	Iris-versicolor
5.8	2.8	4.0	1.0	Iris-versicolor
5.8	2.7	4.8	1.8	Iris-versicolor
5.8	2.6	4.2	1.5	Iris-versicolor
5.7	4.4	1.2	0.2	Iris-setosa
5.7	3.8	5.1	2.4	Iris-virginica
5.7	3.0	4.1	1.0	Iris-versicolor
5.7	2.9	4.0	1.2	Iris-versicolor
5.7	2.8	1.7	0.3	Iris-setosa
5.7	2.6	4.2	1.2	Iris-versicolor
5.7	2.5	4.2	1.3	Iris-versicolor
5.6	3.0	4.5	1.3	Iris-versicolor
5.6	2.9	3.5	1.0	Iris-versicolor
5.6	2.8	5.0	2.0	Iris-virginica
5.6	2.7	4.5	1.5	Iris-versicolor
5.6	2.5	3.6	1.3	Iris-versicolor
5.5	4.2	4.9	2.0	Iris-virginica
5.5	3.5	4.2	1.3	Iris-versicolor
5.5	2.6	3.9	1.1	Iris-versicolor
5.5	2.5	1.4	0.2	Iris-setosa
5.5	2.4	1.3	0.2	Iris-setosa
5.5	2.3	4.4	1.2	Iris-versicolor
5.4	3.9	4.0	1.3	Iris-versicolor
5.4	3.7	3.8	1.1	Iris-versicolor
5.4	3.4	4.0	1.3	Iris-versicolor
5.4	3.0	1.7	0.4	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.2	4.1	1.7	0.2	Iris-setosa
5.2	3.5	4.5	1.5	Iris-versicolor
5.2	3.4	1.5	0.2	Iris-setosa
5.2	2.7	1.5	0.1	Iris-setosa

Table C (c)

sepal-length	sepal-width	petal-length	petal-width	class
5.1	3.8	1.5	0.2	Iris-setosa
5.1	3.7	1.4	0.2	Iris-setosa
5.1	3.5	1.5	0.3	Iris-setosa
5.1	3.4	1.5	0.4	Iris-setosa
5.1	3.3	1.4	0.2	Iris-setosa
5.1	2.5	1.5	0.2	Iris-setosa
5.0	3.6	1.7	0.5	Iris-setosa
5.0	3.5	3.0	1.1	Iris-versicolor
5.0	3.4	1.4	0.2	Iris-setosa
5.0	3.3	1.3	0.3	Iris-setosa
5.0	3.2	1.5	0.2	Iris-setosa
5.0	3.0	1.4	0.2	Iris-setosa
5.0	2.3	1.2	0.2	Iris-setosa
5.0	2.0	1.6	0.2	Iris-setosa
4.9	3.1	3.3	1.0	Iris-versicolor
4.9	3.0	1.5	0.1	Iris-setosa
4.9	2.5	1.4	0.2	Iris-setosa
4.9	2.4	4.5	1.7	Iris-virginica
4.8	3.4	3.3	1.0	Iris-versicolor
4.8	3.1	1.6	0.2	Iris-setosa
4.8	3.0	1.6	0.2	Iris-setosa
4.7	3.2	1.4	0.1	Iris-setosa
4.6	3.6	1.3	0.2	Iris-setosa
4.6	3.4	1.0	0.2	Iris-setosa
4.6	3.2	1.4	0.3	Iris-setosa
4.6	3.1	1.4	0.2	Iris-setosa
4.5	2.3	1.5	0.2	Iris-setosa
4.4	3.2	1.3	0.2	Iris-setosa
4.4	3.0	1.3	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.3	3.0	1.1	0.1	Iris-setosa

Table C (d)

Appendix D: Results of Iris experiment (top 11 points as ranked by DKN)

Original data		
length	width	id
7.9	3.8	1
7.7	3.8	2
7.7	3	3
7.7	2.8	4
7.7	2.6	5
7.6	3	6
7.4	2.8	7
7.3	2.9	8
7.2	3.6	9
7.2	3.2	10
7.2	3	11

k=3					k=4					k=5					k=6				
distance	diff	length	width	id	distance	diff	length	width	id	distance	diff	length	width	id	distance	diff	length	width	id
1.701	0.132	7.9	3.8	1	1.775	0.143	7.9	3.8	1	1.965	0.084	7.9	3.8	1	2.016	0.096	7.9	3.8	1
1.569	0.398	7.7	3.8	2	1.632	0.290	7.7	3.8	2	1.881	0.369	7.7	3.8	1	1.920	0.316	7.7	3.8	2
1.171	0.047	5.7	4.4	59	1.342	0.275	5.7	4.4	59	1.512	0.351	5.7	4.4	59	1.604	0.421	5.7	4.4	59
1.124	0.200	5	2	99	1.067	0.111	5	2	99	1.161	0.019	4.5	2.3	112	1.183	0.049	4.5	2.3	112
0.924	0.030	7.2	3.6	9	0.956	0.028	7.7	3	3	1.142	0.018	5.5	4.2	71	1.134	0.010	7.7	2.6	5
0.895	0.051	7.7	2.6	5	0.928	0.004	7.2	3.6	9	1.124	0.058	5	2	99	1.124	0.000	5.5	4.2	71
0.843	0.015	6.2	2.2	43	0.924	0.000	5.4	3.7	78	1.067	0.106	7.7	2.6	5	1.124	0.116	5	2	99
0.828	0.095	4.5	2.3	112	0.924	0.022	5.5	4.2	71	0.960	0.036	7.2	3.6	9	1.008	0.106	7.2	3.6	9
0.733	0.001	6	2.2	52	0.902	0.000	5.2	4.1	82	0.924	0.037	5.2	4.1	82	0.902	0.000	5.8	4	55
0.732	0.000	5.8	4	55	0.902	0.015	7.7	2.6	5	0.887	0.000	5.8	4	55	0.902	0.000	6.2	2.2	43
0.732	0.022	5.2	4.1	82	0.887	0.000	4.5	2.3	112	0.887	0.059	6.2	2.2	43	0.902	0.008	5.2	4.1	82

k=7					k=10					k=20					k=30				
distance	diff	length	width	id	distance	diff	length	width	id	distance	diff	length	width	id	distance	diff	length	width	id
1.853	0.060	7.9	3.8	1	2.264	0.247	5.7	4.4	59	2.886	0.200	5.7	4.4	59	2.721	0.145	5.7	4.4	59
1.793	0.160	7.7	3.8	2	2.016	0.096	7.9	3.8	1	2.687	0.345	5.5	4.2	71	2.576	0.107	7.9	3.8	1
1.633	0.389	5.7	4.4	59	1.920	0.287	7.7	3.8	2	2.341	0.102	7.9	3.8	1	2.469	0.020	7.7	3.8	2
1.245	0.000	5	2	99	1.633	0.029	5.3	3.7	81	2.240	0.003	5.8	4	55	2.448	0.209	5.5	4.2	71
1.245	0.034	4.5	2.3	112	1.604	0.000	5.4	3.7	78	2.237	0.210	7.7	3.8	2	2.240	0.000	5.8	4	55
1.211	0.040	5.5	4.2	71	1.604	0.140	5.5	4.2	71	2.026	0.147	5.4	3.9	77	2.240	0.023	5	2	99
1.171	0.034	5.4	3.9	77	1.464	0.010	4.5	2.3	112	1.879	0.075	5	2	99	2.217	0.197	5.2	4.1	82
1.137	0.000	7.2	3.6	9	1.454	0.034	7.7	2.6	5	1.804	0.006	5.2	4.1	82	2.020	0.044	5.4	3.9	77
1.137	0.092	7.7	2.6	5	1.420	0.038	5.8	4	55	1.799	0.129	5.7	3.8	60	1.976	0.117	4.5	2.3	112
1.045	0.014	7.7	2.8	4	1.383	0.041	5	2	99	1.670	0.038	4.5	2.3	112	1.858	0.005	7.7	2.6	5
1.031	0.103	7.7	3	3	1.342	0.100	5.2	4.1	82	1.632	0.122	7.7	2.6	5	1.853	0.054	7.2	3.6	9

Notes: (1) “id” column has been added to allow a cross-referencing with different result sets.

(2) “Original data was ranked in terms of distance from origin (0,0). Table shows only first 11 data points.

(3) The “boxed” cell in each diff column shows the threshold value above which points are classed as outliers.

Appendix E:

Detailed DKN results for modified Iris database (k=30)

Sorted list in order of distance from each point to its 30th neighbour

Distance	sepal-length	sepal-width	petal-length	petal-width	class
2.72074	5.7	4.4	1.5	0.4	Iris-setosa
2.57588	7.9	3.8	6.4	2	Iris-virginica
2.46891	7.7	3.8	6.7	2.2	Iris-virginica
2.44844	5.5	4.2	1.4	0.2	Iris-setosa
2.23971	5.8	4	1.2	0.2	Iris-setosa
2.23971	5	2	3.5	1	Iris-versicolor
2.21713	5.2	4.1	1.5	0.1	Iris-setosa
2.02011	5.4	3.9	1.7	0.4	Iris-setosa
1.97579	4.5	2.3	1.3	0.3	Iris-setosa
1.85835	7.7	2.6	6.9	2.3	Iris-virginica
1.85292	7.2	3.6	6.1	2.5	Iris-virginica
1.79872	7.7	3	6.1	2.3	Iris-virginica
1.79313	5.7	3.8	1.7	0.3	Iris-setosa
1.74221	7.7	2.8	6.7	2	Iris-virginica
1.68647	7.6	3	6.6	2.1	Iris-virginica
1.63316	5	2.3	3.3	1	Iris-versicolor
1.60399	5.1	3.8	1.5	0.3	Iris-setosa
1.60399	5.4	3.7	1.5	0.2	Iris-setosa
1.60399	5.3	3.7	1.5	0.2	Iris-setosa
1.57479	4.9	2.4	3.3	1	Iris-versicolor
1.56909	4.3	3	1.1	0.1	Iris-setosa
1.56569	6.2	2.2	4.5	1.5	Iris-versicolor
1.54688	4.4	3.2	1.3	0.2	Iris-setosa
1.50991	4.4	2.9	1.4	0.2	Iris-setosa
1.46379	4.4	3	1.3	0.2	Iris-setosa
1.46228	5.1	3.7	1.5	0.4	Iris-setosa
1.45399	4.6	3.4	1.4	0.3	Iris-setosa
1.45399	6	2.2	4	1	Iris-versicolor
1.44311	4.9	2.5	4.5	1.7	Iris-virginica
1.39377	7.4	2.8	6.1	1.9	Iris-virginica
1.38652	4.6	3.6	1	0.2	Iris-setosa
1.38265	6.3	2.3	4.4	1.3	Iris-versicolor
1.38265	5.5	2.3	4	1.3	Iris-versicolor
1.36198	5.1	2.5	3	1.1	Iris-versicolor
1.35367	4.6	3.1	1.5	0.2	Iris-setosa
1.34704	5	3.6	1.4	0.2	Iris-setosa
1.34137	7.2	3.2	6	1.8	Iris-virginica
1.34137	4.6	3.2	1.4	0.2	Iris-setosa
1.31771	4.8	3.4	1.6	0.2	Iris-setosa
1.29705	7.3	2.9	6.3	1.8	Iris-virginica
1.24464	7.2	3	5.8	1.6	Iris-virginica
1.24464	4.7	3.2	1.3	0.2	Iris-setosa
1.24216	5.5	2.4	3.8	1.1	Iris-versicolor

1.21101	6.7	2.5	5.8	1.8 Iris-virginica
1.21101	5	3.5	1.3	0.3 Iris-setosa
1.18281	6.4	2.9	4.3	1.3 Iris-versicolor
1.17137	5.2	3.5	1.5	0.2 Iris-setosa
1.17137	6.7	3.3	5.7	2.1 Iris-virginica
1.17137	5.5	3.5	1.3	0.2 Iris-setosa
1.17137	6.3	3.4	5.6	2.4 Iris-virginica
1.17137	5.1	3.5	1.4	0.2 Iris-setosa
1.17137	4.8	3	1.4	0.1 Iris-setosa
1.17137	5.8	2.8	5.1	2.4 Iris-virginica
1.16147	6.2	3.4	5.4	2.3 Iris-virginica
1.16147	6	3.4	4.5	1.6 Iris-versicolor
1.14613	7.1	3	5.9	2.1 Iris-virginica
1.14613	7	3.2	4.7	1.4 Iris-versicolor
1.14222	5.7	2.8	4.5	1.3 Iris-versicolor
1.14222	4.8	3.1	1.6	0.2 Iris-setosa
1.13702	6.5	3	5.8	2.2 Iris-virginica
1.13702	5	3.4	1.5	0.2 Iris-setosa
1.13406	4.9	3	1.4	0.2 Iris-setosa
1.13406	4.9	3.1	1.5	0.1 Iris-setosa
1.12436	5.5	2.5	4	1.3 Iris-versicolor
1.11835	6.3	3.3	4.7	1.6 Iris-versicolor
1.06899	5	3.3	1.4	0.2 Iris-setosa
1.06663	6.6	3	4.4	1.4 Iris-versicolor
1.06663	5.1	3.3	1.7	0.5 Iris-setosa
1.06663	6.9	3.2	5.7	2.3 Iris-virginica
1.06663	5.1	3.4	1.5	0.2 Iris-setosa
1.05417	5.2	2.7	3.9	1.4 Iris-versicolor
1.05417	5.6	2.5	3.9	1.1 Iris-versicolor
1.04532	6.7	3	5	1.7 Iris-versicolor
1.04532	5.4	3.4	1.7	0.2 Iris-setosa
1.04532	5	3.2	1.2	0.2 Iris-setosa
1.04532	5	3	1.6	0.2 Iris-setosa
1.03125	6.8	3.2	5.9	2.3 Iris-virginica
1.03125	5.2	3.4	1.4	0.2 Iris-setosa
1.03125	6.9	3.1	4.9	1.5 Iris-versicolor
1.00811	6.1	2.6	5.6	1.4 Iris-virginica
1.00811	5.5	2.6	4.4	1.2 Iris-versicolor
0.967419	6.6	2.9	4.6	1.3 Iris-versicolor
0.967419	5.7	2.6	3.5	1 Iris-versicolor
0.967419	6	2.7	5.1	1.6 Iris-versicolor
0.967419	6.3	2.5	4.9	1.5 Iris-versicolor
0.967419	5.7	2.5	5	2 Iris-virginica
0.960137	5.8	2.7	4.1	1 Iris-versicolor
0.955719	5.4	3	4.5	1.5 Iris-versicolor
0.955719	6.8	2.8	4.8	1.4 Iris-versicolor
0.955719	5.9	3.2	4.8	1.8 Iris-versicolor
0.929177	6.8	3	5.5	2.1 Iris-virginica
0.927971	6.1	2.8	4	1.3 Iris-versicolor
0.927971	5.6	3	4.5	1.5 Iris-versicolor
0.927971	5.7	3	4.2	1.2 Iris-versicolor
0.927971	6.4	3.2	4.5	1.5 Iris-versicolor
0.924344	5.8	2.6	4	1.2 Iris-versicolor
0.924344	6.5	3.2	5.1	2 Iris-virginica
0.902187	6.4	2.7	5.3	1.9 Iris-virginica
0.902187	5.6	2.7	4.2	1.3 Iris-versicolor

0.902187	5.6	2.9	3.6	1.3	Iris-versicolor
0.89468	5.7	2.9	4.2	1.3	Iris-versicolor
0.89468	6.5	2.8	4.6	1.5	Iris-versicolor
0.89468	6.1	3	4.6	1.4	Iris-versicolor
0.887417	6.3	2.7	4.9	1.8	Iris-virginica
0.887417	6.7	3.1	4.4	1.4	Iris-versicolor
0.887417	6.2	2.8	4.8	1.8	Iris-virginica
0.887417	5.6	2.8	4.9	2	Iris-virginica
0.843235	6.4	2.8	5.6	2.1	Iris-virginica
0.828105	6.3	2.8	5.1	1.5	Iris-virginica
0.828105	5.9	3	4.2	1.5	Iris-versicolor
0.816148	6.1	2.9	4.7	1.4	Iris-versicolor
0.816148	6.4	3.1	5.5	1.8	Iris-virginica
0.756086	6	3	4.8	1.8	Iris-virginica
0.756086	6.2	2.9	4.3	1.3	Iris-versicolor
0.733053	6.3	2.9	5.6	1.8	Iris-virginica
0.731897	6	2.9	4.5	1.5	Iris-versicolor

Likely Outliers:
These are assumed to have large distance to kth neighbour. Threshold is determined by point of largest change in list of distances - approx. derivative.
The distance prior to which the largest change occurs is: 2.23971
4 records are likely outliers.

2.72074	5.7	4.4	1.5	0.4	Iris-setosa
2.57588	7.9	3.8	6.4	2	Iris-virginica
2.46891	7.7	3.8	6.7	2.2	Iris-virginica
2.44844	5.5	4.2	1.4	0.2	Iris-setosa

Appendix F: Error localisation experiments

F.1 UK Annual Business Inquiry (Retail Section)

The ABI dataset consists of a sample from the retail section of the U.K. Annual Business Inquiry (1999). For each business the dataset contains information on aspects of purchase costs and employment costs. The data is based on responses for either of 2 possible questionnaires. The larger questionnaire contains 17 retail questions. The smaller one only asks for summary information and contains 5 retail questions. Businesses on the dataset therefore either have 5 or 17 responses to individual questions.

It was necessary to create a special "errors only" dataset from the ABI datasets provided to Euredit partners. This is because the Euredit datasets have not been provided in "errors only" versions, but only as combined "missings plus errors" versions. The outlier detection system under development at York is currently designed to perform imputation and outlier detection separately. These operations could ultimately be combined, but the aim of research at York is to evaluate performance on these tasks separately, so that the results of imputation are not dependent on the results of outlier (or error) detection and vice-versa.[2]

Initial results are based on the sec297xxx.csv data files provided for Euredit. A derived dataset was created by comparing a clean data file: sec297(true).csv with a corresponding dirty file containing errors and missing values: sec297(y3).csv, by replacing any missing value records in the dirty file with the equivalent correct record from the clean file. Two new files were created:

- a new version of sec297(y3).csv containing only errors (no records with missings) called: 297_dirty.csv

- a reference file containing only those records from sec297(y3).csv which are known to contain errors called: 297_errs_only.csv

In addition, the long and short response records were considered separately and together, to investigate the possibility that performance might be adversely affected by the mixture of higher and lower dimensional data. More data files were created for this purpose:

- two more versions of sec297(true).csv the clean data file: 297short_clean.csv and 297long_clean.csv

- two more versions of 297_dirty.csv: 297short_dirty.csv and 297_dirty_long.csv

These files provide a framework for evaluating some basic performance characteristics of the York system, where any records identified as potential outliers can be checked against the file containing known errors. Nevertheless, evaluating the performance of the York outlier/error detection system is currently difficult, because it is aimed at identifying and ranking outliers at the *record level* (rather than at the level of individual values). The York system is currently configured to output a list of outlier candidates ranked in descending order of confidence. It may be possible to extend the York system in future, to identify

[2] Of course the combined performance may be of *practical* interest, but it is essential to know the performance of individual components if it is desired to combine two different methods, or if only one component is needed for a particular application. Hopefully these issues will be revisited and addressed in the Euredit project.

individuals or groups of outlier values, but the level of confidence is likely to be quite low, and probably inadequate for a completely automated system. In practice it would seem preferable to identify potential records-with-errors and leave it to the user to make more specific decisions about what errors are actually present.

Another problem affecting evaluation is the difficulty in identifying which of the known errors are "soft errors" and which are "hard errors", particularly since facilities to implement edit rules are not available at York. Additional metadata produced by ISTAT has proved invaluable, though the methods used to produce "filtered data" at York are quite basic and very time-consuming. Since other partners seem likely to need various forms of "filtered data" it would appear sensible to make additional dataset versions available to all partners.

The results shown in the following series of graphs are thus limited due to difficulties in preparing data in a suitable format, and for this reason the current results do not necessarily reflect a complete range of experiments, but rather a selected set of examples which provide some basic information. The experimental procedure used was as follows:

- set the chosen parameters
- train the system using a data file containing records with and without errors (but not missings)
- use the DKN strategy described previously to rank each record in the data file and store in output file
- calculate results by working down the ranked records, from high to low, checking whether each corresponds with a genuine error (true positive) or not (false positive)
- this last step is carried out in steps of size 10, so for example, the first 10 ranked records are examined to obtain a score out of 10 for true positives and false positives, then the first 20, 30, etc.

For each experiment the essential system parameters recorded are:

- **MaxBins** – the maximum number of quantisation bins used by the RUE algorithm,
- **k** – the value of k used to find the k nearest neighbours
- **CMMmatches** – the number of matches required from CMM matching. This is always larger than k and is used to try to ensure that the true (Euclidean) nearest neighbours are found.

Approximate processing time (total elapsed time) is also noted for completeness, although the system used to generate results contains many inefficiencies due residual "debug" code used during development which is still to be removed.

Note that in the following series of bar graphs, the numbers shown for each successive bar are *cumulative*, effectively showing the scores for true positive and false positive out of the total possible up to that point (e.g. scores out of 10, 20, 30, etc.).

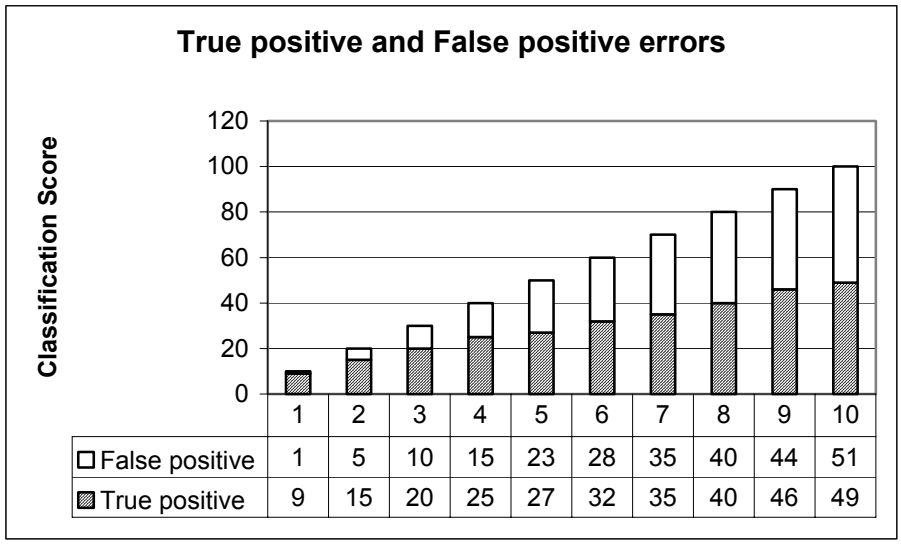
F.1.1 ABI sector 297 (long version) – all error types

These experiments examined the effect of different parameter settings on the ability of the system to correctly identify errors of all types, but using only the subset of records representing responses to the *long* version of the questionnaire. This is of interest because the responses to the short version contain fewer values and hence exist in only a subspace of the long version responses. This difference in information content may adversely affect the operation of the DKN method. The charts below are from earlier experiments where scores are shown only for the first 100 classifications. Later charts show scores for the first 200 classifications.

Number of data records in file was: 1177

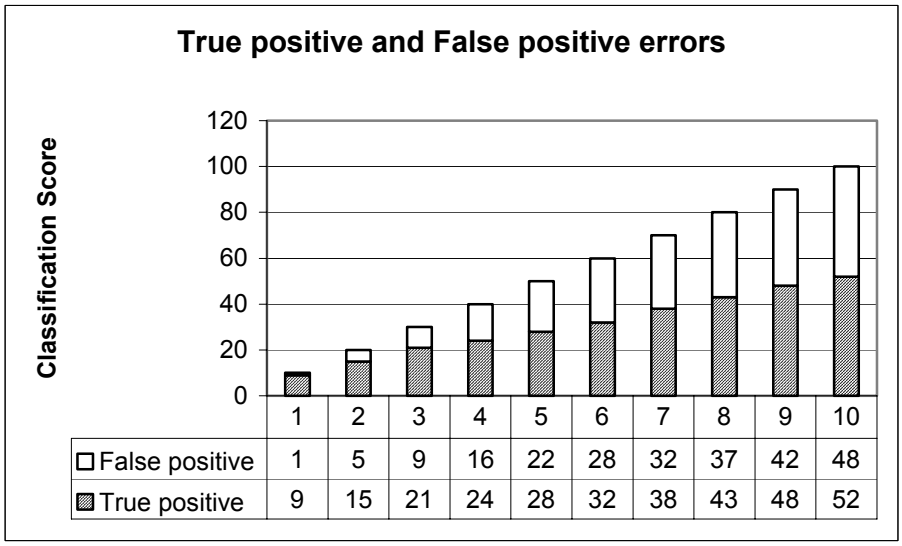
Processing time on Dell laptop (433MHz Celeron) was around (TBC) minutes.

a) k = 100, MaxBins = 34, CMMmatches = 300



	1	2	3	4	5	6	7	8	9	10
False positive	1	5	10	15	23	28	35	40	44	51
True positive	9	15	20	25	27	32	35	40	46	49

b) k = 100, MaxBins = 34, CMMmatches = 400



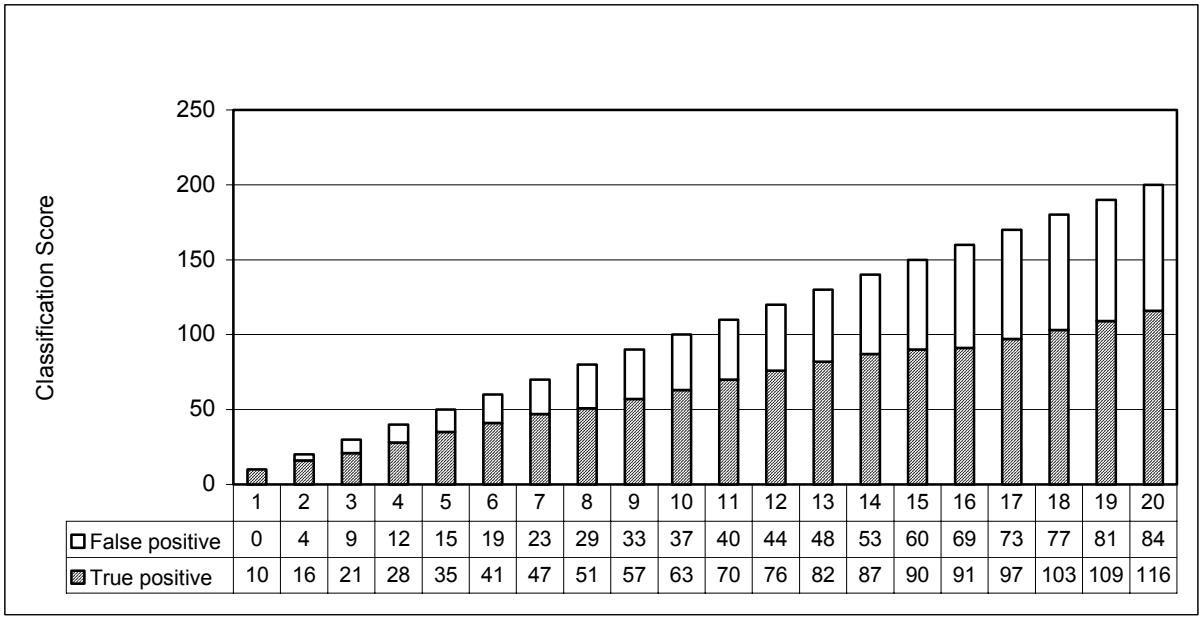
	1	2	3	4	5	6	7	8	9	10
False positive	1	5	9	16	22	28	32	37	42	48
True positive	9	15	21	24	28	32	38	43	48	52

F.1.2 ABI sector 297 (short version) – all error types

This examines the ability of the system to correctly identify errors of all types, but using only the subset of records representing responses to the *short* version of the questionnaire. This is of interest because the responses to the short version contain fewer values and hence exist in only a subspace of the long version responses. This difference in information content may adversely affect the operation of the DKN method.

k = 100, MaxBins = 56, CMMmatches = 500, Number of data records: 3,149

Processing time on Dell laptop (433MHz Celeron) was around 16.5 minutes.



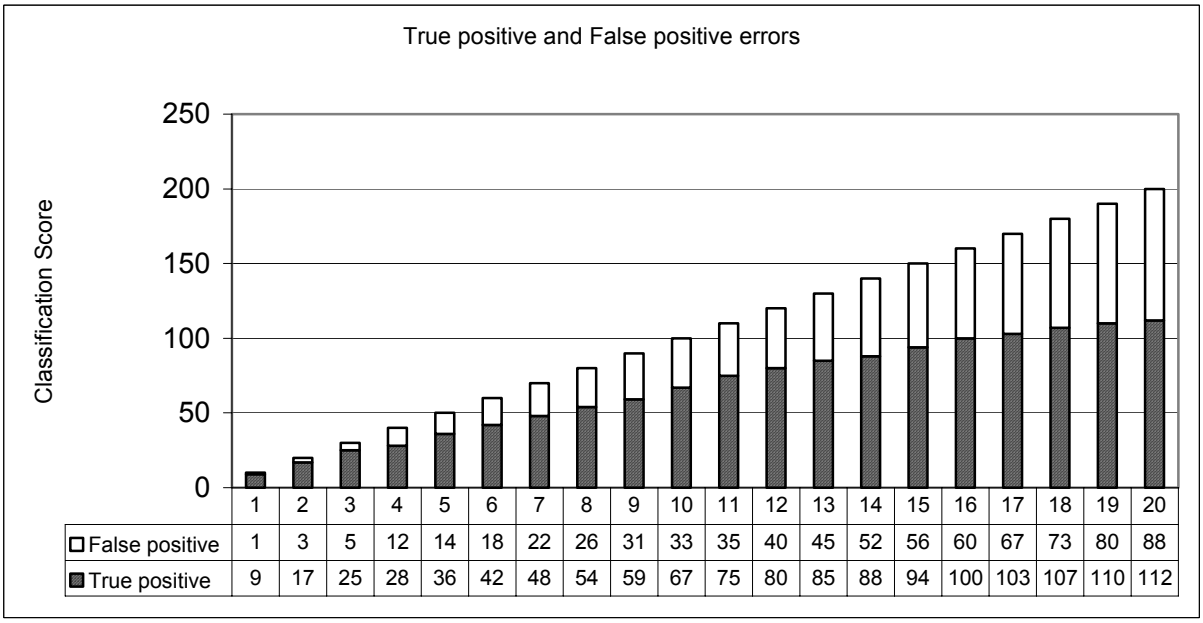
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
False positive	0	4	9	12	15	19	23	29	33	37	40	44	48	53	60	69	73	77	81	84
True positive	10	16	21	28	35	41	47	51	57	63	70	76	82	87	90	91	97	103	109	116

F.1.3 ABI sector 297 (combined short and long version) – all error types

This examines the ability of the system to correctly identify errors of all types, using records representing responses to both the long and short versions of the questionnaire.

k = 100, MaxBins = 65, CMMmatches = 500, Number of data records: 4,325

Processing time on Dell laptop (433MHz Celeron) was around 23.3 minutes.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
False positive	1	3	5	12	14	18	22	26	31	33	35	40	45	52	56	60	67	73	80	88
True positive	9	17	25	28	36	42	48	54	59	67	75	80	85	88	94	100	103	107	110	112

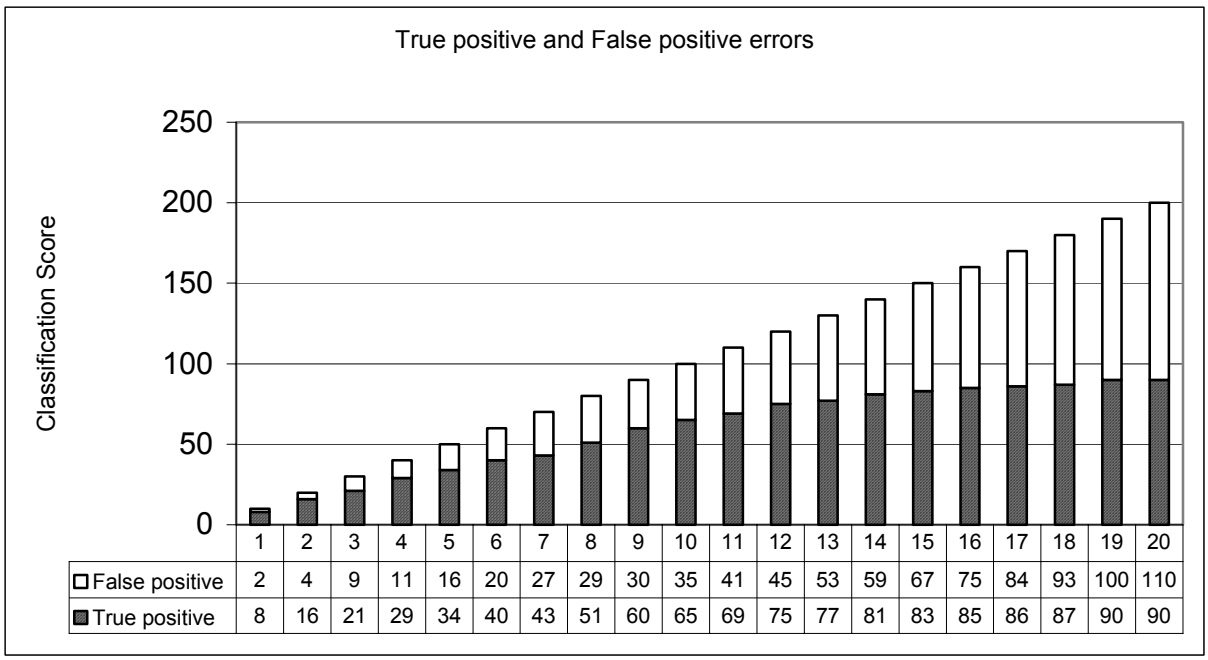
F.1.4 ABI sector 297 (combined short and long version) – soft errors only

This series examines the effect of different parameter settings on the ability of the system to correctly identify soft errors only (hard edit failures have been removed from the data and replaced with "clean" unperturbed versions. The system is particularly designed with the aim of finding errors of this general type. Records representing responses to both the long and short versions of the questionnaire are present.

The experiment is repeated for selected values of k: 100, 10, 5, and 1.

k = 100, MaxBins = 65, CMMmatches = 500, Number of data records: 4,325

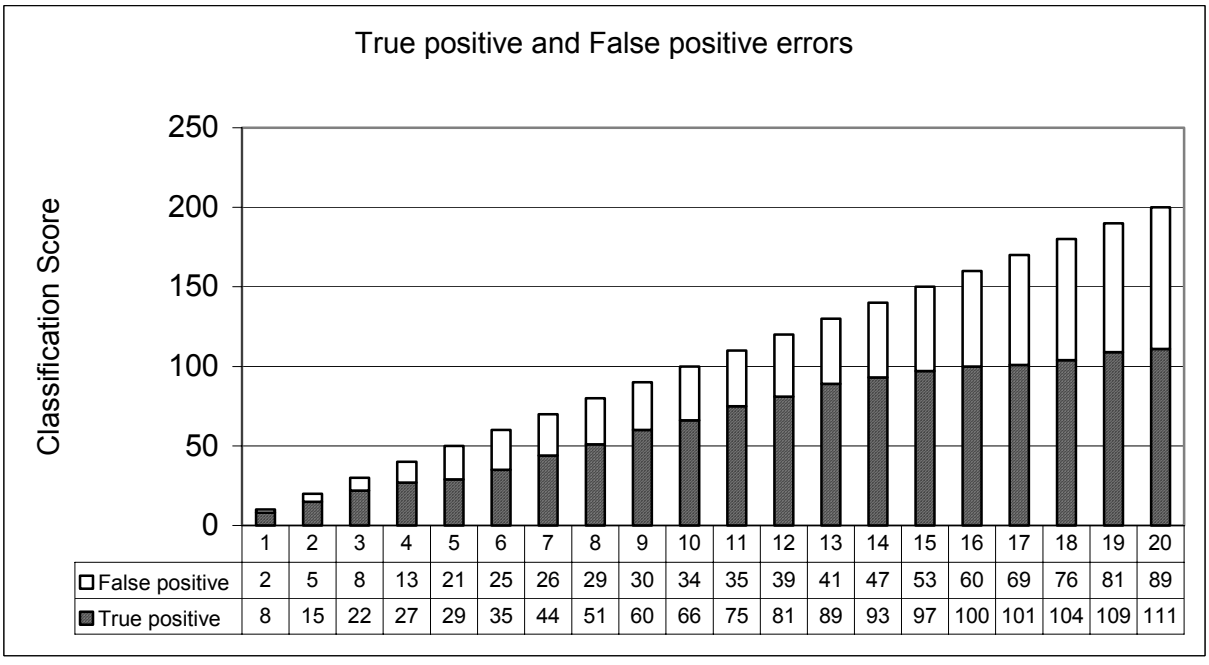
Processing time on Dell laptop (433MHz Celeron) was around 24 minutes.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
False positive	2	4	9	11	16	20	27	29	30	35	41	45	53	59	67	75	84	93	100	110
True positive	8	16	21	29	34	40	43	51	60	65	69	75	77	81	83	85	86	87	90	90

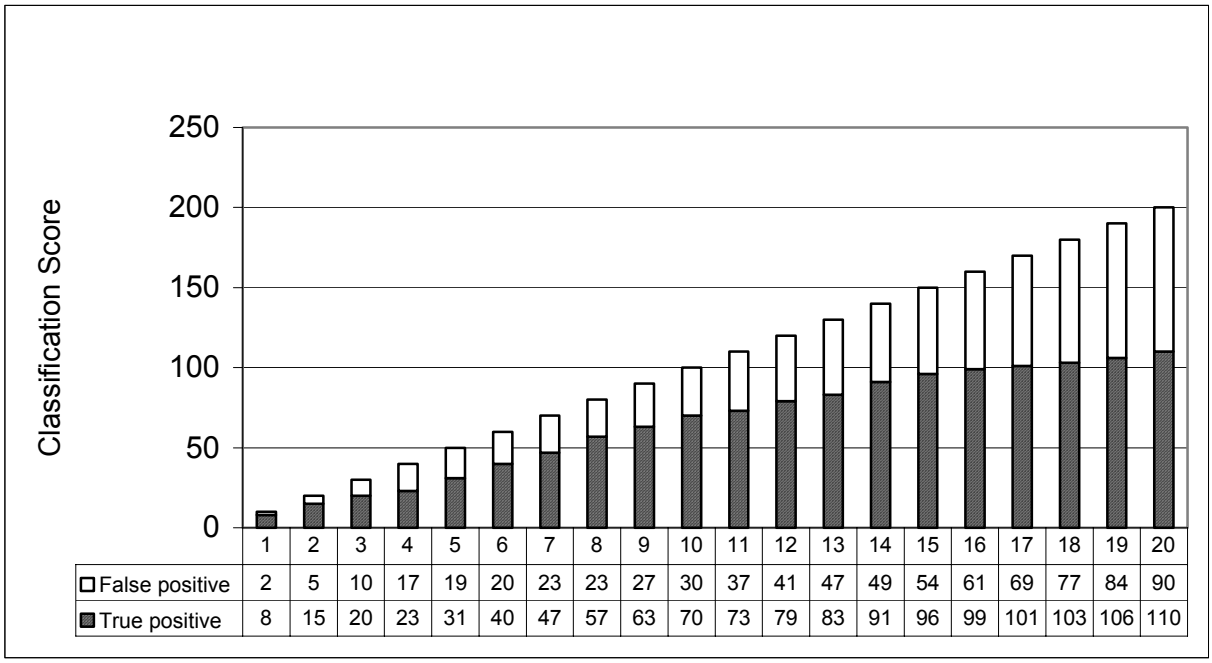
k = 10, MaxBins = 65, CMMmatches = 100, Number of data records: 4,325

Processing time on Dell laptop (433MHz Celeron) was around 5.1 minutes.



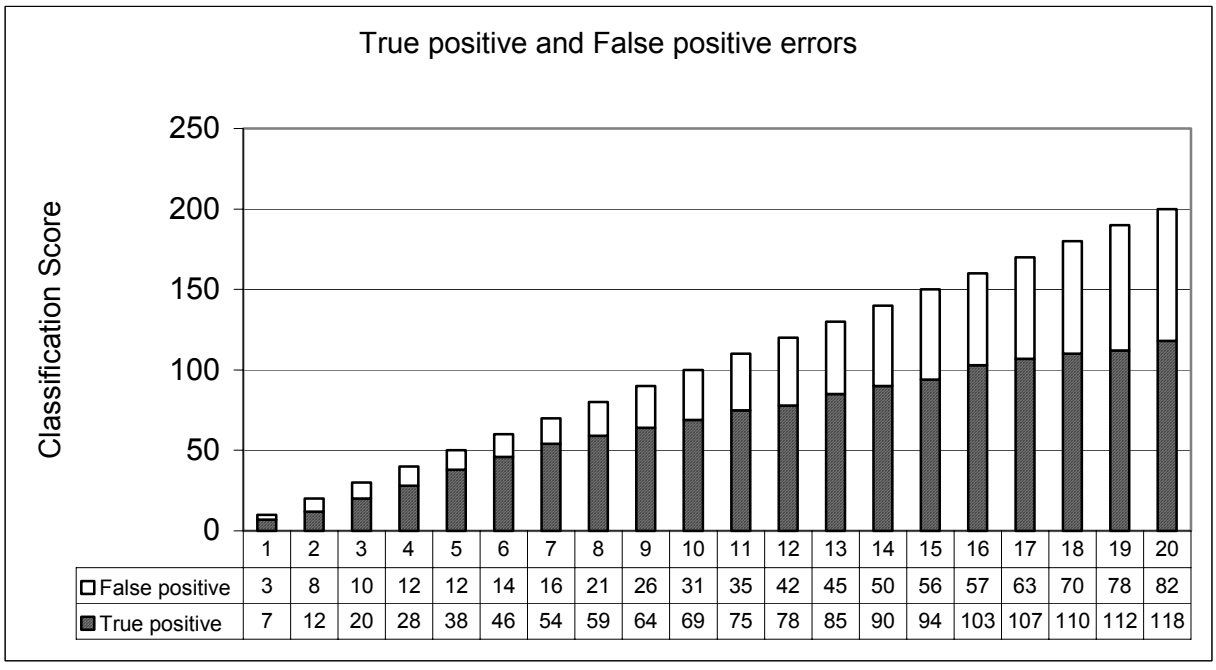
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
False positive	2	5	8	13	21	25	26	29	30	34	35	39	41	47	53	60	69	76	81	89
True positive	8	15	22	27	29	35	44	51	60	66	75	81	89	93	97	100	101	104	109	111

k = 5, MaxBins = 65, CMMmatches = 100, Number of data records in file was: 4,325
Processing time on Dell laptop (433MHz Celeron) was around 4.25 minutes.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
False positive	2	5	10	17	19	20	23	23	27	30	37	41	47	49	54	61	69	77	84	90
True positive	8	15	20	23	31	40	47	57	63	70	73	79	83	91	96	99	101	103	106	110

k = 1, MaxBins = 65, CMMmatches = 100, Number of data records in file was: 4,325
Processing time on Dell laptop (433MHz Celeron) was around 2.75 minutes.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
False positive	3	8	10	12	12	14	16	21	26	31	35	42	45	50	56	57	63	70	78	82
True positive	7	12	20	28	38	46	54	59	64	69	75	78	85	90	94	103	107	110	112	118