Hierarchical Single Label Classification: An Alternative Approach

Esra'a Alshdaifat, Frans Coenen and Keith Dures

Abstract In this paper an approach to multi-class (as opposed to multi-label) classification is proposed. The idea is that a more effective classification can be produced if a coarse-grain classification (directed at groups of classes) is first conducted followed by increasingly more fine-grain classifications. A framework is proposed whereby this scheme can be realised in the form of a classification hierarchy. The main challenge is how best to create class groupings with respect to the labels nearer the root of the hierarchy. Three different techniques, based on the concepts of clustering and splitting, are proposed. Experimental results show that the proposed mechanism can improve classification performance in terms of average accuracy and average AUC in the context of some data sets.

1 Introduction

Classification is concerned with the creation of a global model to be used for predicting the class labels of new data. Classification can be viewed as a three-step process: (i) generation of the classifier using appropriately formatted "training" data, (ii) testing of the effectiveness of the generated classifier using test data and (iii) application of the classifier. The first two steps are sometimes combined for experimental purposes. There exist many possible models for classifier generation; the classifier can, for example, be expressed in terms of rules, decision trees or mathematical formula.

E. Alshdaifat (✉) · F. Coenen · K. Dures
Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom
e-mail: esraa@liv.ac.uk

F. Coenen
e-mail: coenen@liv.ac.uk

K. Dures
e-mail: dures@liv.ac.uk

M. Bramer and M. Petridis (eds.), *Research and Development in Intelligent Systems XXX*,
DOI: 10.1007/978-3-319-02621-3_3,

The performance of classifiers can also vary greatly according to the nature of the input data, how the input data is preprocessed, the adopted generation mechanism and number of classes in the data set. To date no one classification model has been identified that is, in all cases, superior to all others in terms of classification effectiveness [9]. In an attempt to improve classifier performance ensemble models have been proposed. An ensemble model is a composite model comprised of a number of classifiers. There is evidence to suggest that ensembles are more accurate than their individual component classifiers [11].

Classification can be categorised according to: (i) the size of the set of classes C from which class labels may be drawn, if $|C| = 2$ we have a simple "yes-no" (binary) classifier, if $|C| > 2$ we have a multi-class classifier, and (ii) the number of class labels that may be assigned to a record (single-label classification versus multi-label classification), for most applications we typically wish to assign a single label to each record. The work described in this paper is directed at multi-class single-label classification, especially where $|C|$ is large. Multi-class classification is challenging for two reasons. The first is that the training data typically used to generate the desired classifier often features fewer examples of each class than in the case of binary training data. The second reason is that it is often difficult to identify a suitable subset of features that can effectively serve to distinguish between large numbers of classes (more than two class labels). The "Letter Recognition" data set [1], frequently used as a benchmark dataset with respect to the evaluation of machine learning algorithms, is an example of a data set with a large number of class labels; the data set describes black-and-white rectangular pixel displays each representing one of the twenty six capital letters available in the English alphabet.

Three most straightforward approaches for solving multi-class classification problems include: (i) using stand-alone classification algorithm such as decision tree classifiers [14], and Bayesian classification [12], (ii) building a sequence of binary classifiers [17], and (iii) adopting an ensemble approach such as bagging [3] or boosting [10]. The solution proposed on this paper is to adopt an ensemble approach founded on the idea of arranging the classifiers into a hierarchical form. Class hierarchy models are usually directed at multi-label classification where we have multiple-class sets (as opposed to single-label classification); each level in the hierarchy is directed at a specific class set. In our case the proposal is that the hierarchy be directed at single-label classification. Classifiers at the leaves of our hierarchy feature binary classifiers, while classifiers at nodes further up the hierarchy feature classifiers directed at groupings of class labels. The research challenge is how best to organise (group) the class labels so as to produce a hierarchy that generates an effective classification. A number of alternative techniques are proposed in this paper, whereby this may be achieved, founded on ideas concerned with clustering and splitting mechanisms to divide the class labels across nodes.

The rest of this paper is organised as follows. Section 2 gives a review of related work on multi-class classification. Section 3 describes the proposed hierarchical classification approach. Section 4 presents an evaluation of the proposed hierarchical classification approach as applied to a range of different data sets. Section 5 summarizes the work and indicates some future research directions.

2 Literature Review

In this section we review some of the previous work related to the work described in this paper. The section is organised as follows: we first (Sect. 2.1) consider classification algorithms that can be directly used to address multi-class classification problems; then we go on to (Sect. 2.2) consider ensemble mechanisms in the context of the multi-class classification problem.

2.1 Using Stand-Alone Classification Algorithms to Solve Multi-Class Classification Problems

Some classification algorithms are specifically designed to address binary classification, for example support vector machines [18]. However, such algorithms can be adapted to handle multi-class classification by building sequence of binary classifiers. Other classification algorithms can directly handle any number of class labels; examples include: decision tree classifiers [14], Classification Association Rule Mining (CARM) [7] and Bayesian classification [12]. Among these decision trees algorithms are of interest with respect to the work described in this paper because it can be argued that our hierarchies have some similarity with the concept of decision trees. Decision trees have a number of advantages with respect to some other classification techniques: (i) they can be constructed relatively quickly, (ii) they are easy to understand (and modify) and (iii) the tree can be expressed as a set of "decision rules" (which is of benefit with respect to some applications). Decision trees are constructed by inducing a split in the training data according to the values associated with the available attributes. The splitting is frequently undertaken according to "Information Gain" [14], "Gini Gain" [5], or "Gain Ratio" [14]. Each leaf node of a decision tree holds a class label. A new example is classified by following a path from the root node to a leaf node, the class held at the identified leaf node is then considered to be the class label for the example [11]. Amongst the most frequently quoted decision tree generation algorithms are: ID3 [14], C4.5 [15] and CART [5].

2.2 Using Ensemble Classifiers to Solve Multi-Class Classification Problems

The simplest form of ensemble can be argued to comprise a suite of binary classifiers trained using n different binary training sets, where n is the total number of class labels to be considered, each one trained to distinguish the examples in a single class from the examples in all remaining classes [17]. This scheme is often referred to as the One-Versus-All (OVA) [16] scheme. The one-versus-all scheme is conceptually simple, and has been independently proposed by numerous researchers [16].

A variation of the one-versus-all scheme is All-Versus-All (AVA) [17] where a classifier is constructed for every possible pair of classes. Given n classes this will result in $(n(n-1)/2)$ classifiers. To classify a new example, each classifier "votes", and the new example is assigned the class with the maximum number of votes. AVA tends to be superior to OVA [11]. The problem with this type of classification is that binary classifiers are typically sensitive to errors; if a classifier produces an erroneous result it can adversely affect the final vote count. In order to improve on this type of binary classification; it is possible to introduce "error-correcting output codes", which change the definition of the class a single classifier has to learn [8].

A more sophisticated form of ensemble is one where the classifiers are arranged in concurrent (parallel) [3] or sequential (serial) [10] form. In the concurrent ensemble methodology, the original dataset is partitioned into several subsets from which multiple classifiers are induced concurrently. The simplest approach, often referred to as "Bagging" (the name was obtained from the phrase "Bootstrap Aggregation") [13], is where "sampling with replacement" is used [3]. A well-known bagging algorithm is the "Random Forest" algorithm [4], which combines the output from a collection of decision trees.

In the sequential case there is interaction between the different classifiers (the outcome of one feeds into the next). Thus it is possible to take advantage of knowledge generated in a previous iteration to guide the learning in the next iterations. One well studied form of sequential ensemble classification is known as "Boosting", where a sequence of weak classifiers are "chained" together to produce a single composite strong classifier in order to achieve a higher combined accuracy than that which would have been obtained if the weak classifiers were used independently. A well-known boosting algorithm is Adaboost [10] .

Regardless of how an ensemble system might be configured, an important issue is how results are combined to enhance classification accuracy. The simplest approach is to use some kind of voting system [2]. Voting algorithms can be divided into two types: those that adaptively change the distribution of the training set based on the performance of previous classifiers (as in boosting methods) and those that do not (as in Bagging).

3 The Hierarchical Single-Label Classification Framework

As noted in the introduction to this paper the proposed hierarchical classification mechanism is a form of ensemble classifier. Each node in our hierarchy holds a classifier. Classifiers at the leaves conduct fine-grained (binary) classifications while the classifiers at non-leaf nodes further up the hierarchy conduct coarse-grained classification directed at categorising records using groups of labels. An example hierarchy is presented in Fig. 1. At the root we classify into two groups of class labels $\{a, b, c, d\}$ and $\{e, f, g\}$. At the next level we split into smaller groups, and so on till we reach classifiers that can associate single class labels with records. Note that Fig. 1 is just an example of the proposed hierarchical model; non-leaf child nodes

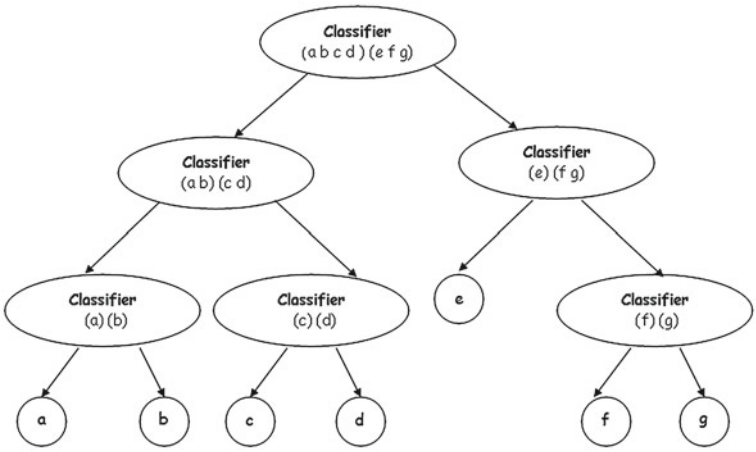

Fig. 1 Classification hierarchy example

may end up with overlapping classifications due to the result returned from the used clustering algorithms.

The challenge of hierarchical single-label classification, as conceived in this paper, is how best to distribute groupings of class labels at non-leaf nodes in the class hierarchy. A number of different mechanisms are considered based on clustering and splitting criteria: (i) k-means clustering, (ii) hierarchical clustering, and (iii) data splitting. The motivation for using ensemble classifiers arranged in a hierarchical form so as to improve the accuracy of the classification were thus: (i) the established observation that ensemble methods tend to improve classification performance [11], and (ii) dealing with smaller subsets of class labels at each node might produce better results. The remainder of this section is organised as follows. Section 3.1 explains the generation of the hierarchical model, while Sect. 3.2 illustrates its operation.

3.1 Generating the Hierarchical Model

In the proposed model classifiers nearer the root of the hierarchy conduct coarse-grain classification with respect to subsets of the available set of classes. Classifiers at the leaves of the hierarchy conduct fine-grain (binary) classification. To create the hierarchy a classifier needs to be generated for each node in the hierarchy using an appropriately configured training set. The process is illustrated in Fig. 1 where the root classifier classifies the data set into two sets of class labels $\{a, b, c, d\}$ and $\{e, f, g\}$, the level two classifiers are then directed at further subsets and so on. The sets of class labels (the label groupings) are identified by repeatedly dividing the data using one of the proposed clustering or splitting approaches.

The proposed hierarchy generation algorithm is presented in Algorithm 1. The algorithm assumes a data structure, called *hierarchy*, comprised of the following fields:

1. Classifier: A classifier (root and body nodes only, set to null otherwise).
2. Left: Reference to left branch of the hierarchy (root and body nodes only, set to null otherwise).
3. Right: Reference to right branch of the hierarchy (root and body nodes only, set to null otherwise).

Considering the algorithm presented in Algorithm 1 in further detail. The algorithm in this case assumes a binary tree (hierarchy) structure (this is not necessarily always the case but the assumption allows for a more simplified explanation with respect to this paper). The *Generate_Hierarchy* procedure is recursive. On each recursion the input to the *Generate_Hierarchy* procedure is the data set D (initially this is the entire training set). If the number of classes represented in D is one (all the records in D are of the same class) a leaf hierarchy node will be created (labeled with the appropriate class label). If the number of classes featured in D is two, a binary classifier is constructed (to distinguish between the two classes). The most sophisticated part of the *Generate_Hierarchy* procedure is the following option, which applies if the number of classes featured in D is more than two. In this case the records in D are divided into two groups $D1$ and $D2$ each with a meta-class label, $K1$ and $K2$, associated with it. A Classifier is then constructed to discriminate between $K1$ and $K2$. The *Generate_ Hierarchy* procedure is then called again, once with $D1$ (representing the left branch of the hierarchy) and once with $D2$ (representing the right branch of the hierarchy).

Two types of classifiers were considered with respect to the nodes within our hierarchy: (i) straight forward single "stand-alone" classifiers (Fig. 2a), and (ii) bagging ensembles (Fig. 2b). With respect to the first approach a simple decision tree classifier was generated for each node in the hierarchy. With respect to bagging, the data set D associated with each node was randomly divided into N disjoint partitions

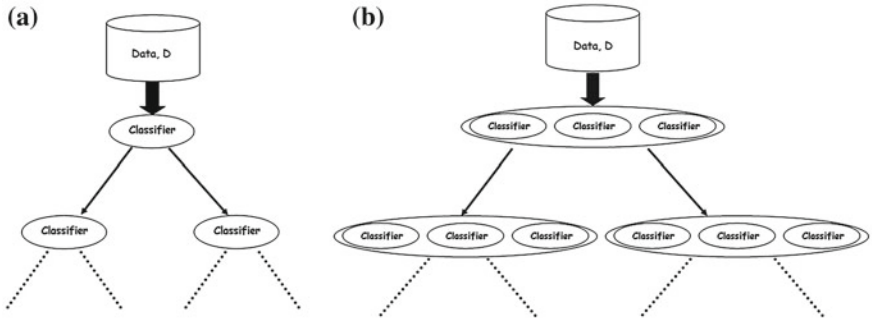

Fig. 2 Hierarchical classification, **a** using a single classifier at each node and **b** using a Bagging ensemble at each node

Algorithm*: Generate_ Hierarchy*
Input:

1. Data partition, D, which is a set of training tuples and their associated class labels.
2. Classification method, a procedure for building *classifier* at a given hierarchy node.
3. Clustering (or splitting) method, a procedure for splitting the classes between nodes.

Output: A hierarchical classifiers.

Method:

```
create a hierarchy node N;
if number of classes featured in D is one class, C, then
    return N as a leaf node labeled with C,
    ( assign null value to hierarchy left, and right);
else
    if number of classes featured in D is two classes then
        create a hierarchy classifier (using classification method), to distinguish between the
        two (real) classes (binary classifier);
        assign null value to hierarchy left, and right;
    else
        cluster D into two clusters K1 and K2 (using clustering techniques);
        recast labels in D so that they correspond to K1 and K2;
        create a hierarchy classifier (using classification method), to distinguish between K1
        and K2;
        D1 = records in D containing class labels in K1;
        hierarchy right= Generate_ Hierarchy (D1);
        D2 = records in D containing class labels in K2;
        hierarchy left = Generate_ Hierarchy (D2);
    end
end
return N;
```

Algorithm 1: Ensemble hierarchy generation algorithm

and a classifier generated for each (in the evaluation reported in Sect. 4, $N = 3$ was used).

As already noted, with respect to dividing D during the hierarchy generation process, three different techniques were considered: (i) k-means, (ii) hierarchical clustering, and (iii) data splitting. Of these k-means is the most well-known and commonly used partitioning method where the input is divided into k partitions (in our model $k = 2$ was used because of the binary nature of our hierarchy used for experimental purposes). Hierarchical clustering creates a hierarchical decomposition of the given data. In the context of the work described in this paper a divisive hierarchical clustering approach (top-down) was used because this fits well with respect to our vision of hierarchical ensemble classification. This process commences with all records in one cluster, in each successive iteration, a cluster is split into smaller clusters until a "best" cluster configuration is arrived at (measured using cluster cohesion and separation measures). Data splitting comprises a simple "cut" of the data into two groups so that each contains a disjoint subset of the entire set of class labels.

3.2 Classification Using Hierarchical Model

Section 3.1 explained the generation of the desired hierarchical model. After the model has been generated, the intention is to use it to classify new unseen data records. In this section the process whereby this is achieved is explained. Recall from the above that during the generation process sets of class labels are grouped. For simplicity, and in acknowledgement of the binary nature of our example hierarchies, we refer to these groups as the $left$ and $right$ groups. The procedure for using the hierarchy to classify a record, R, is summarized in Algorithm 2. The $Classifying\_Records\_Using\_Hierarchy$ procedure is recursive. On each recursion the algorithm is called with two parameters: (i) R, the record to be classified, and (ii) a pointer to the current node location in the hierarchy (at start this will be the root node). How the process proceeds then depends on the nature of the class label returned by the classifier at the current node in the hierarchy. If the returned class belongs to one of either the $left$ or $right$ groups $Classifying\_Records\_Using\_Hierarchy$ will be called again with the parameters: (i) either the left or right child node as appropriate, and of course (ii) the record R. If the returned class label is a specific class label (as opposed to some grouping of labels) this class label will be returned as the label to be associated with the given record and the algorithm terminated.

Algorithm: *Classifying_Records_Using_Hierarchy*
Input:

1. New unseen record, R.
2. Hierarchy node, N. (pointer to the hierarchy)

Output: The predicted class label of the input record R

Method:
use node's *classifier* to classify R;
if the returned class label is member of original (real) class labels **then**
  return class label;
else
  if the returned class label is member of right class labels group **then**
    Classifying_Records_Using_Hierarchy (R, node N right branch);
  else
    Classifying_Records_Using_Hierarchy (R, node N left branch);
  end
end

Algorithm 2: Ensemble hierarchy classification algorithm

4 Experimentation and Evaluation

In this section we present an overview of the adopted experimental set up and the evaluation results obtained. The experiments used 14 different data sets (with different numbers of class labels) taken from UCI data repository [1], which were processed

using the LUCS-KDD-DN software [6]. Ten-fold Cross Validation (TCV) was used throughout. The evaluation measures used were average accuracy and average AUC (Area Under the receiver operating Curve).

The following individual experiments were conducted and the results recorded:

1. Decision Trees (DT): Stand-alone classification of the data using a single classifier (no hierarchy), the objective being to establish a bench mark.
2. Bagging like strategy (Bagging): Classification of the data using a bagging ensemble classifier comprised of three classifiers (no hierarchy), the objective was to establish a second benchmark.
3. K-means and Decision tree (K-means&DT): The proposed approach using k-means ($k = 2$) to group data with decision tree classifiers at each node.
4. Data splitting and Decision tree (DS&DT): The proposed approach using data splitting to group data with decision tree classifiers at each node.
5. Hierarchical clustering and Decision tree (HC&DT): The proposed approach using divisive hierarchical clustering to group data with decision tree classifiers at each node.
6. K-means and Bagging (K-means&B): The proposed approach using k-means ($k = 2$) to group data with a bagging ensemble classifier at each node.
7. Data splitting and Bagging (DS&B): The proposed approach using data splitting to group data with a bagging ensemble classifier at each node.
8. Hierarchical clustering and Bagging (HC&B): The proposed approach using divisive hierarchical clustering to group data with a bagging ensemble classifier at each node.

With respect to the bagging methods three decision tree classifiers were generated with respect to each node.

The results in terms of average accuracy are presented in Fig. 3 in the form of a collection of Fourteen histograms, one for each data set considered. The best method (from the above list) is indicated to the top right of each histogram (in some cases two best methods were identified). From the figure it can be observed that the proposed hierarchical techniques can significantly improve the classification accuracy with respect to eight out of the Fourteen data sets considered. In two out of these eight cases (where hierarchical classification worked well) bagging produced the best result; in the remaining cases the decision tree classifiers produced the best result.

Figure 4 shows the results in terms of the average AUC of the eight techniques considered in the evaluation (again the best result with respect to each dataset is indicated). From the figure it can be observed that the proposed hierarchical techniques can enhance the classification AUC with respect to eight of the Fourteen data sets considered. In the remaining six cases out of the Fourteen the stand-alone decision tree classifier produced the best AUC result (although in one case the same AUC result was the same as that produced using our hierarchical approach and in another case the same as that produced using bagging).

It is interesting to note that the proposed hierarchical techniques work well for datasets with high numbers of class labels (18 in the case of the Chess KRvK dataset and 15 in the case of the Soybean dataset). The reason for this is that a high number

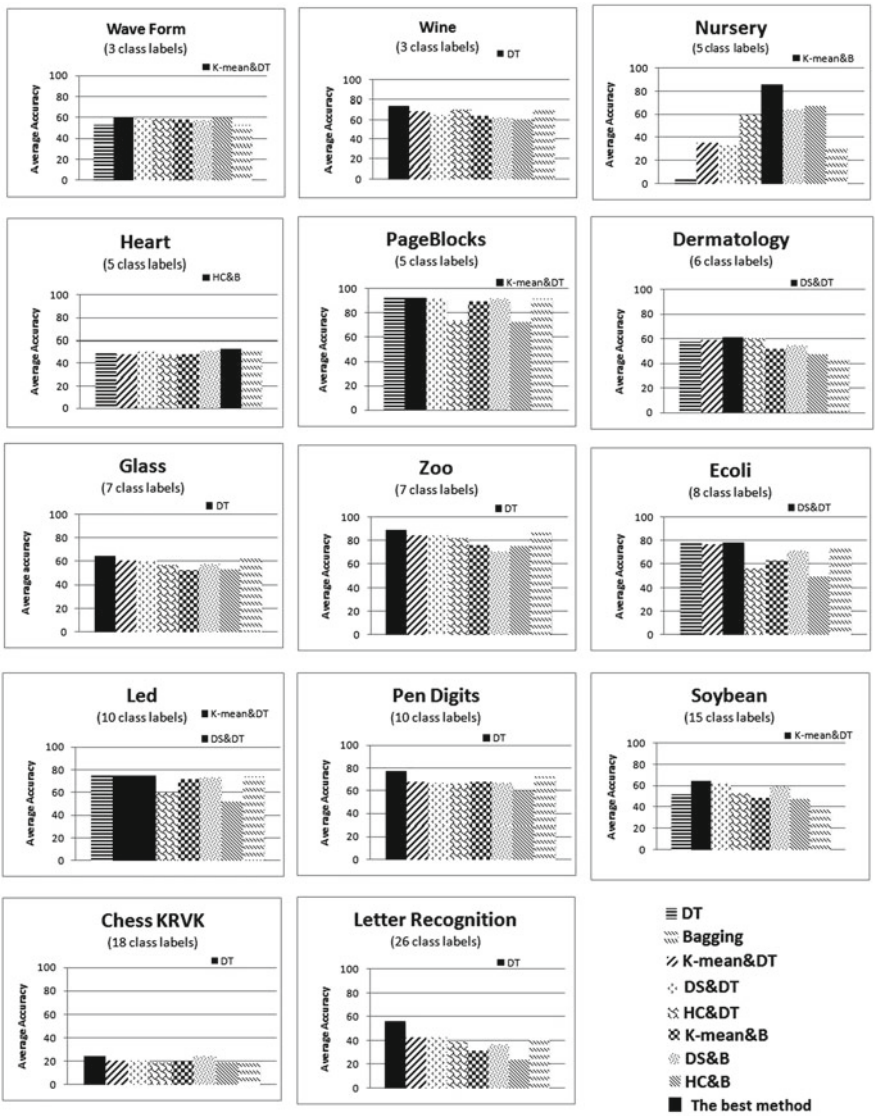

Fig. 3 Average accuracy for the Fourteen different evaluation data sets

of class labels allow for the generation of more sophisticated hierarchies, while a low number of classes does not. Thus data sets with high numbers of class labels are deemed to be much more compatible with the proposed approach.

The raw data on which the histograms presented in Figs. 3 and 4 were based is presented in Tabular form in Table 1 (the best result with respect to each dataset is highlighted).

Table 1 Average accuracy and AUC for the Fourteen different evaluation data sets

Data set	Classes	DT		Bagging		K-mean&DT		DS&DT		HC&DT		K-mean&Bagging	DS&B		HC&B	
		ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
WaveForm	3	53.72	0.54	53.36	0.53	59.68	0.60	58.56	0.59	58.98	0.59	58.08	0.58	57.48	0.58	59.44	0.59
Wine	3	73.86	0.73	69.91	0.69	68.75	0.67	65.22	0.64	70.54	0.68	64.05	0.64	62.45	0.60	60.09	0.59
Nursery	5	5.15	0.03	32.71	0.16	35.69	0.19	33.51	0.27	59.70	0.32	86.20	0.46	64.55	0.32	67.32	0.36
Heart	5	48.80	0.28	51.15	0.28	47.97	0.31	49.89	0.32	47.48	0.26	47.41	0.24	51.42	0.28	52.86	0.25
PageBlocks	5	92.55	0.49	92.23	0.47	92.56	0.49	92.40	0.48	74.28	0.36	89.77	0.20	91.56	0.28	72.88	0.20
Dermatology	6	57.53	0.57	43.95	0.39	58.90	0.55	61.37	0.60	60.25	0.54	51.90	0.43	54.54	0.49	47.30	0.42
Glass	7	64.50	0.40	62.27	0.36	60.44	0.39	60.91	0.38	56.95	0.34	52.10	0.28	57.43	0.28	52.90	0.25
Zoo	7	89.00	0.53	87.27	0.53	85.00	0.50	85.00	0.50	82.09	0.51	76.27	0.44	71.36	0.41	75.45	0.43
Ecoli	8	78.07	0.34	73.61	0.31	76.90	0.33	78.72	0.35	56.29	0.28	63.59	0.23	71.63	0.23	49.44	0.22
Led	10	74.72	0.74	74.06	0.74	75.00	0.75	75.00	0.75	59.94	0.59	71.75	0.72	73.22	0.73	51.75	0.51
PenDigits	10	76.84	0.77	72.64	0.72	67.30	0.67	66.78	0.67	66.30	0.66	67.22	0.67	66.74	0.67	59.44	0.59
Soybean	15	52.12	0.52	37.18	0.30	64.62	0.65	63.02	0.56	52.67	0.47	48.74	0.38	59.06	0.49	48.01	0.39
ChessKRVK	18	24.05	0.13	18.20	0.09	20.45	0.13	20.50	0.15	18.80	0.13	19.85	0.10	23.95	0.11	18.95	0.10
LetRecog	26	56.05	0.56	41.82	0.42	42.99	0.43	42.76	0.43	38.59	0.39	31.12	0.31	36.92	0.37	24.26	0.24
Mean		60.50	0.47	57.88	0.43	61.16	0.48	60.97	0.48	57.35	0.44	59.15	0.41	60.17	0.42	52.86	0.37
Standerd Deviation		24.00	0.22	21.64	0.20	19.45	0.19	19.58	0.17	15.53	0.17	19.43	0.19	16.31	0.18	15.92	0.16

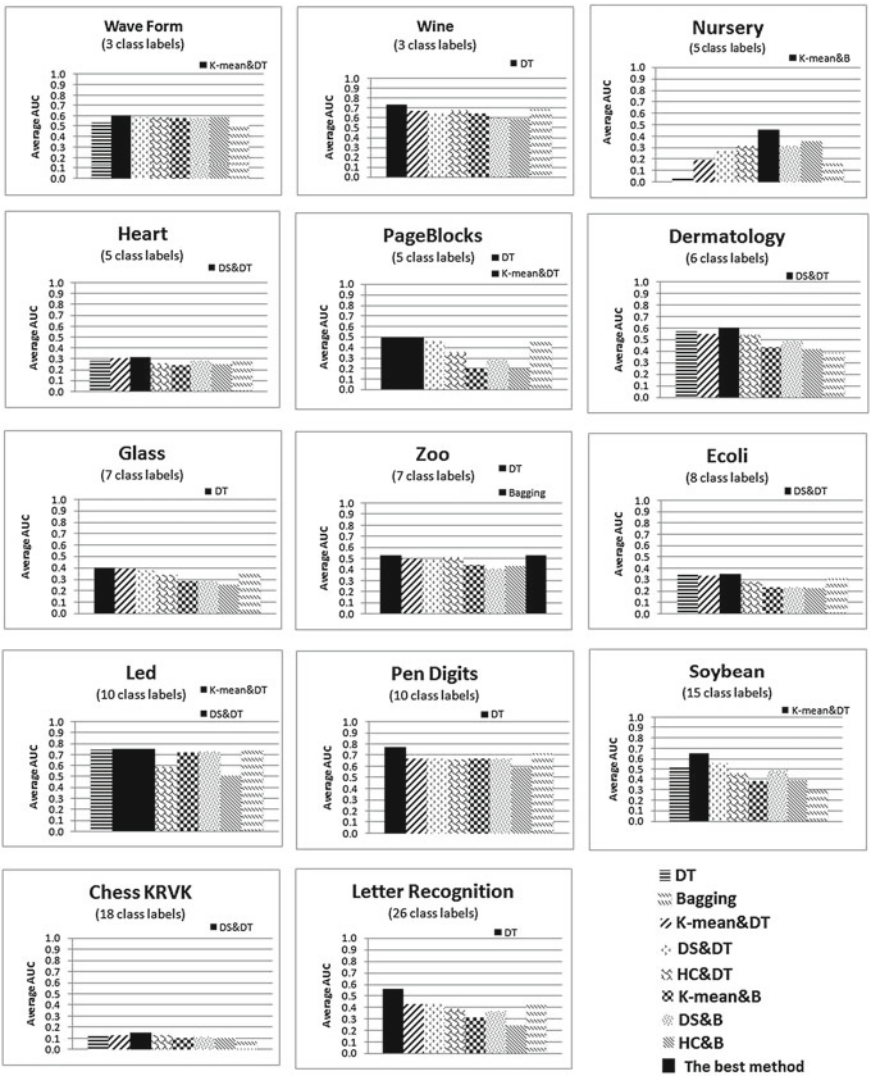

Fig. 4 Average AUC for the Fourteen different evaluation data sets

5 Conclusions

A classification technique to achieve multi-class classification using the concept of a classification hierarchy has been described. The idea is to conduct the classification starting in a coarse-grain manner, where records are allocated to groups, proceeding to a fine-grain manner until class labels can be assigned. To generate such a hierarchical classifier three different grouping techniques were considered: (i) *k*-mean,

(ii) hierarchical clustering and (iii) data splitting. We also considered the use of two different styles of classifier at each node: (i) decision tree classifiers and (ii) bagging ensemble classifiers.

From the evaluation presented in this paper it was demonstrated that the proposed hierarchical classification model could be successfully used to classify data in a more effective manner than when stand-alone classifiers were used. Each of the proposed methods tended to improve the accuracy and/or AUC of the classification process with respect to some of the data sets considered. Best results were produced using k-mean and decision tree classifiers (K-mean&DT), and data splitting and decision tree classifiers (ST&DT). An issue with the proposed approach is that if a record is miss classified early on in the process (near the root of the hierarchy) there is no opportunity for recovery, regardless of the classifications proposed at lower level nodes and the final leaf nodes. There is therefore no opportunity whereby the proposed approach can attempt to correct its self. To address this issue a number of alternatives will be considered for future work. The first of these is to use classification algorithms, such as Association Rule Mining Classification (CARM) or Bayesian classification, where a confidence or probability value will result. Consequently more than one branch in the tree can may be followed as a consequence of the nature of this value, and a final decision made according to some accumulated confidence or probability value. The use of other tree structures, than binary tree structures, will also be considered; again to allow for the possibility of exploring more than one alternative at each node. One way of achieving this is to use a Directed Acyclic Graph (DAG) structure, instead of a tree hierarchy structure so as to include a greater number of possible class label combinations at each level.

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
2. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning pp. 105–139 (1999).
3. Breiman, L.: Bagging predictors. Machine Learning (2), 123–140 (1996).
4. Breiman, L.: Random forests. In. Machine Learning, pp. 5–32 (2001).
5. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA (1984).
6. Coenen, F.: The LUCS-KDD discretised/normalised arm and carm data library (2003). http://www.csc.liv.ac.uk/frans/KDD/Software/LUCS_KDD_DN
7. Coenen, F., Leng, P.: The effect of threshold values on association rule based classification accuracy. Journal of, Data and Knowledge Engineering (2), pp 345–360 (2007).
8. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. JAIR (1995).
9. Dunham, M.H.: Data Mining:Introductory and Advanced Topics. Prentice Hall (2003).
10. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence (5), 771–80 (1999).
11. Jiawei, H., Micheline, K., Jian, P.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2011).
12. Leonard, T., Hsu, J.S.: Bayesian Methods: An Analysis for Statisticians and Interdisciplinary Researchers. Cambridge University Press (2001).

13. Machov, K., Bark, F., Bednr, P.: A bagging method using decision trees in the role of base classifiers (2006).
14. Quinlan, J.R.: Induction of decision trees. Machine Learning (1), 81–106 (1986).
15. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993).
16. Rifkin, R.M., Klautau, A.: In defense of one-vs-all classification. Journal of Machine Learning Research pp. 101–141 (2004).
17. Tax, D.M.J., Duin, R.P.W.: Using two-class classifiers for multiclass classification. In: ICPR (2), pp. 124–127 (2002).
18. Vapnik, V.N.: The Nature of Statistical Learning Theory. Statistics for Engineering and Information Science. Springer (2000).