A Directed Acyclic Graph (DAG) Ensemble Classification Model: An Alternative Architecture for Hierarchical Classification

Esra'a Alshdaifat, Frans Coenen, Keith Dures

Abstract

In this paper a hierarchical ensemble classification approach, that utilizes a Directed Acyclic Graph (DAG) structure, is proposed as a solution to the multi-class classification problem. Two main DAG structures are considered: (i) rooted DAG, and (ii) non-rooted DAG. The main challenges that are considered in this paper are: (i) the successive misclassification issue associated with hierarchical classification, and (i) identification of the starting node within the non-rooted DAG approach. To address these issues the idea is to utilize Bayesian probability values to: select the best starting DAG node, and to dictate whether single or multiple paths should be followed within the DAG structure. The reported experimental results indicated that the proposed DAG structure is more effective than when using a simple binary tree structure for generating a hierarchical classification model.

Keywords: Hierarchical Classification, Ensemble, Directed Acyclic Graph (DAG), Multi-class Classification.

Introduction

A recognized issue associated with Single-label Multi-class classification, where examples are associated with exactly one element of the set of class labels C, $C > 2$, is that when the number of class labels $|C|$ increases the effectiveness of the classification tends to diminish. The Ensemble methodology is considered to be one of the most effective strategies to handle the multi-class problem (Bauer & Kohavi, 1999; Dietterich, 2000; Hansen & Salamon, 1990; Jiawei et al., 2011; Opitz & Maclin, 1999; Oza & Tumer, 2008; Quinlan, 1996; Zhou, 2009). The ensemble model is a composite model comprised of a number of learners (classifiers), often referred to as base learners or weak learners, that "collaborate" to obtain a better classification performance than can be obtained from using a single stand-alone model. Classifiers making up an ensemble can be arranged in three main formats: (i) concurrent (Breiman, 1996, 2001; Machov et al., 2006), (ii) sequential (Freund et al., 1999; Wirth &Catlett, 1988), and (iii) hierarchical (Athimethphat & Lerteerawong, 2012; Chen et al., 2004; Kumar et al., 2002; Lei & Govindaraju, 2005; Madzarov et al., 2008). A commonly adopted structure for hierarchical model generation is a binary tree constructed in either a bottom-up or a top-down manner (Beygelzimer et al., 2007; Kumar et al., 2002).

The novel idea presented in this paper is the generation and usage of a hierarchical ensemble classification model that involves arranging the base classifiers in the form of Directed Acyclic Graph (DAG) structure, where each node in the DAG holds a classifier. Nodes near the root of the hierarchy hold classifiers designed to discriminate between groups of class labels while the leaves hold classifiers designed to distinguish between individual class labels. So as to distribute class labels across the DAG nodes the use of a "combination technique" is proposed.

One of the most significant advantages of the DAG classification approach, compared to (say) the binary tree approach, is the ability to include a greater number of possible class label combinations at each level. The work presented in this paper considers two alternative DAG structures to support the generation of the desired DAG hierarchical classification approach: (i) rooted DAG, and (ii) non-rooted DAG. The rooted DAG structure is the most straightforward; however, as will become apparent later in this paper, it entails a number of disadvantages in the context of scalability, effectiveness, and efficiency. The proposed non-rooted structure seeks to address the disadvantages of the rooted DAG. The features provided by the non-rooted DAG structure are that: (i) it enables the elimination of the root node where the largest number of class combinations are considered, as well as reducing the overall number of levels in the desired model (depth pruning); and (ii) it enables the application of breadth pruning, reducing the number of classifiers that need to be generated at each DAG level so as to reduce the overall size of the DAG further (note that breadth pruning can not be applied to the rooted DAG approach because the rooted DAG requires inclusion of all classes combinations). An issue with respect to the non-rooted DAG structure, as the name implies, is the need to determine the "starting node" (the root) from which the classification process is to commence. To this end it is proposed that a classifier generator, such as Naive Bayes classification, that produces probability values that can be utilized to determine the starting node is used.

The presented work is also concerned with addressing the general drawback of hierarchical classification, that of successive misclassification whereby, if a record is misclassified early on in the process it will continue to be misclassified at deeper levels, regardless of the classifications proposed at lower level nodes and the final leaf nodes. To address this problem a multiple path strategy is proposed (facilitated by the probability values generated by the Naive Bayes classifiers at the DAG nodes). The proposed multi-class DAG hierarchical ensemble model is fully described in this paper. Its operation is evaluated by comparing it with: "stand alone" classification, established ensemble classifiers, and a hierarchical binary tree structure based approach.

Literature Review

This section provides a generic overview of the hierarchical ensemble methodology for solving the multi-class classification problem. The hierarchical ensemble methodology is a relatively recently proposed approach to address the multi-class classification problem which involves the generation of a hierarchical "meta-algorithm" (Kumar et al., 2002; Madzarov et al., 2008). A common structure adopted for hierarchical classification, as noted in the previous section, is a binary tree structure constructed in either a bottom-up or top-down manner (Beygelzimer et al., 2007; Kumar et al., 2002). In the top-down approach, the root node contains the complete set of class labels $\{c_1, c_2, \ldots, c_n\}$. Starting from the root, the set of class labels at each node is recursively split, and a classifier is trained to distinguish between the two subsets. Using the bottom-up approach a merging process is adopted similar to agglomerative hierarchical clustering. The two nodes with the closest distance are merged to form a node describing a new meta-class (Beygelzimer et al., 2007). An example binary tree hierarchy is presented in Figure 1. At the root we discriminate between two groups of class labels $\{a, b, c\}$ and $\{d, e\}$. At the next level we distinguish between smaller groups, and so on, till we reach nodes with classifiers that can assign a single class label to a given record.

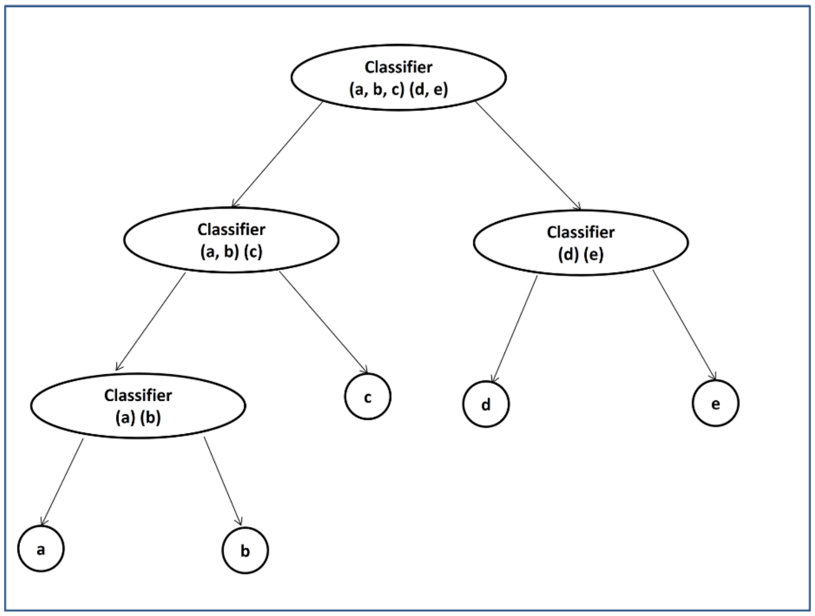

Figure 1. Binary Tree classifier example.

When considering hierarchical classification models the necessary class partitioning can be conducted using a variety of methods such as data splitting or clustering. The performance of the binary tree approach, the most commonly used hierarchical ensemble model, is significantly influenced by the adopted class partitioning method; inappropriate choices can result in poor performance (Alshdaifat, Coenen, & Dures, 2013a, 2013b, 2014). Other than the nature of the grouping method to be adopted, a second drawback of the binary tree based hierarchical ensemble model is that if a record is misclassified early on in the classification process (near the root of the hierarchy) it will continue to be misclassified at deeper levels; the so called "successive misclassification" problem. In previous work the authors have suggested a multiple-path strategy, which allows for more than one path to be followed within the binary tree during the classification stage. This multiple-path strategy was facilitated by the use of classifiers, such as Naive Bayes or Classification Association Rule Mining (CARM), which feature probability or confidence values that can be used to determine where one path should be followed and where two paths should be followed. However, the multi-path strategy only partially resolves the successive misclassification problem, fundamentally the binary tree structure is not sufficiently expressive to capture the nature of multi-class classification.

A more recent work has utilised DAG structures to solve the multiclass classification problem (Songsiri, Phetkaew & Kijsirikul, 2015). More specifically the DAG structure used to combine the prediction results obtained from a set of binary classifiers, which can be considered a special case of using a set of binary classifiers to solve the multi-class classification problem. While with respect to the work presented in this paper, groups of class labels are considered at each DAG node not two classes (binary classification), this will become more apparent later in this paper.

The Rooted Directed Acyclic Graph (rootedDAG) Classification Model Framework

This section describes the proposed rooted DAG classification model. The rooted DAG classification model is founded on the idea of arranging the classifiers into a hierarchical form utilizing a rooted DAG structure. Each node in the rooted DAG classification model holds a classifier. Classifiers at leaves act as binary classifiers while the remaining classifiers (at the root and intermediate nodes) are directed at groupings of class labels. Naive Bayes classifiers are generated at each DAG node, the reason for using this type of classifier is because the probability values produced can be used to direct the hierarchical classification process. In order to group (partition) the input data *D* during the hierarchy generation process, combination techniques are used (as opposed to clustering or splitting as used in the context of binary structure based hierarchical classification systems). The intuition behind using combination techniques to distribute classes between nodes within the DAG is that it could result in a better classification accuracy than the cluster grouping techniques previously considered by the authors, especially because of the large number of class combinations that need to be included at each individual level in the DAG structure. An example rooted DAG classifier, for four class labels C = {*a, b, c, d*}, is presented in Figure 2. At the root we classify into four class groups, all possible combination of size |*C*|-1, at the next level we classify into three class groups. Classifiers at the final level discriminate between two classes; as a result a single class label can be assigned to each record.

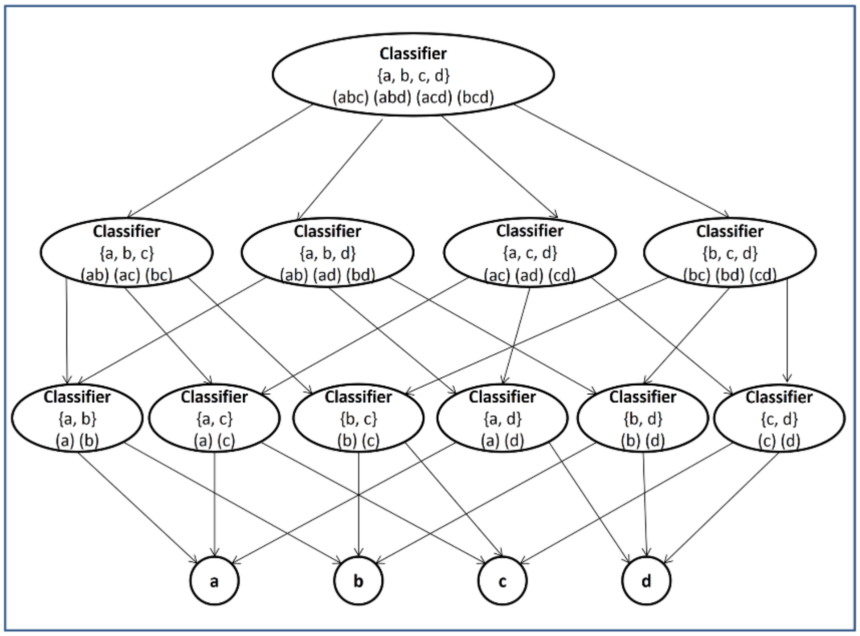

Figure 2. Rooted DAG example.

The following sub-sections explain the generation and operation of the rooted DAG classification model in more detail.

Rooted DAG Generation

To generate the desired *rooted* DAG classification model a Naive Bayes classifier was generated at each DAG node, and combination techniques were utilized to distribute class labels between nodes within the DAG. More specifically, starting at the root of the DAG with the complete class set C (level $i = 0$), the class groupings at each level are identified by finding all possible class combinations of size $|C| - i$ (where i is the level number). As the process proceeds i is increased by one and consequently the "combination size" is decreased by one. The process continues until the combination size reaches two. The number of classifiers that need to be learned in order to generate the DAG classification model can be calculated using (1).

$$\textit{Number Of Classifiers} = 2^{N} - N - 1 \qquad (1)$$

Where N is the number of class labels in a given dataset.

Algorithm 1 presents the generation process in more detail. The input to the algorithm is the training data set D and the set of class labels C. The DAG is created in a recursive manner using the function *dagGen*. On each recursion the *dagGen* function is invoked with two parameters: *combinationSize*, the combination size (starting with $|C|$ - 1 to 2); and *CurrentNodes*, a reference to the current level nodes (resulting from the previous iteration, initially *CurrentNodes* = *root*). The recursive process starts by finding the set of size *combinationSize* class combinations, the set *combinationsSet* (line 12). After that we go through this set (line 14) and on each iteration: (i) a new DAG node, *newNode*, is created (line 15), (ii) the set of training set records that feature the class combination is identified (line 16), (iii) a classifier is trained using the training set records (line 17); and (iv) the new node is added to the set of accumulated level nodes so far, *NodeSet* (line 18). Then we loop through the set of current nodes and add a link from each current node *CurrentNode* to the new node *newNode* whenever the set of class labels associated with the new node is included in the set of class labels associated with a current node. The recursive process terminates if *combinationSize* reaches 2 (line 25).

Rooted DAG Operation

In this section the operation of the rooted DAG classification model is explained. For classifying a new record the most straightforward strategy is to follow a "path" from the root node, according to the classification at each hierarchy node, until a node holding a classifier that can assign a single class label is reached, thus a single path strategy. A disadvantage of the single path strategy is that it is susceptible to the successive misclassification issue discussed earlier. The multiple path strategy seeks to address this issue by using the probability values associated with the Naive Bayes classifiers to decide, at each node, whether to follow single or multiple paths.

Single path strategy. Starting with the single path strategy, Algorithm 2 summarizes the procedure. The classification process is done in a recursive manner using the *dagClassify* procedure. The *dagclassify* procedure is called with two parameters: (i) *r*, the record to be classified; and (ii) *Node*, a pointer to the current node location in the DAG. On each recursion the record *r* is classified using the classifier at the current DAG node (line 10). The process proceeds depending on the nature of the returned class label. If it is a single class label then we return this class (line 12). If we have a group of class labels, *dagClassify* is called again (line 15) with *r* and a pointer (*ChildNode*) to the child node associated with the identified class group. If only a single path is followed within the rooted DAG then *N - 1* classifiers will be evaluated (where *N* is the number of class labels in a given dataset).

Algorithm 1. Rooted DAG Generation

```
1.  Input:
2.  D = the input training dataset
3.  C = the set of Classes featured in D
4.  Output: The generated DAG
5.  Start
6.  combinationSize = |C|−1
7.  root = the root node for the DAG
8.  create root classifier
9.  dagGen(combinationSize, root)
10. End
11. function dagGen(combinationSize, currentNodes)
12.   combinationsSet = set of all class combinations of size combinationSize in C
13.   NodeSet = set of new nodes, initially NodeSet ={}
14.   for each class combination in the combinationSet do
15.       create new node (newNode)
16.       generate newNode training records according to class combinations
17.       generate newNode classifier using training records
18.      add newNode to NodeSet
19.    for each node in the currentNodes do
20.        if newNode class set is subset of currentNode class set then
21.           Add newNode to the currentNode as a child
22.        end if
23.    end for
24.   end for
25.   if k > 2 then
26.     dagGen(combinationSize − 1, NodeSet)
27.   end if
28. end function
```

Multiple paths strategy. The Multiple-Path strategy is designed to address the successive misclassification issue, discussed earlier, that is associated with hierarchical classification. In the multiple-path strategy more than one path can be followed within the DAG classification model. More specifically, the Bayesian probability *P* associated with individual class groups will be used to dictate whether one or more paths will be followed, at each node, according to a predefined threshold sigma σ ($0 \leq \sigma < 1$). Although many paths can be followed at each DAG node, only two paths are suggested as a maximum, at each DAG node, so that comparisons can be made with the binary tree hierarchical ensemble model (where only a maximum of two paths can be followed at each tree node). A second reason is to limit the complexity of the proposed DAG model, the need for this will become clear later in this paper in the evaluation section where the classification time is reported for single and multiple path strategies. An issue associated with the suggested multiple-path strategy is how to decide the final class label from the collection of “candidate classes” resulting from following multiple paths. Several mechanisms can be adopted, such as: (i) applying some voting scheme and selecting the candidate class associated with the highest vote, or (iii) generating an accumulated weight for each candidate class and selecting the class associated with the highest accumulated weight. According to previous work conducted by the authors (Alshdaifat, Coenen, & Dures, 2013b, 2014)the last strategy is likely to produce the best classification performance, thus it is adopted with respect to the work presented in this paper. Using this strategy we take into consideration all probability values in a followed path to produce an accumulated value. More specifically, the probability values for a followed path are summed and then divided by the number of classifiers used in the path to produce a *NormalisedAccumulatedProbability* value, (0 < *Normalised Accumulated Probability* <1). The normalized accumulated probability value is calculated for each candidate class, the candidate class associated with the highest value will be retrieved as the class label for a given record. Because of space limitations the Multi-Path procedure is included in Algorithm 4. In the worst case the number of classifiers to be evaluated is given by:

$$\textit{Number Of Classifiers} = 2^{(N-1)} - 1 \qquad (2)$$

Where N is the number of class labels in a given dataset.

Algorithm 2. Rooted DAG single path Classification

```
1.  Input:
2.  r = A new unseen record
3.  Root = Start node for the DAG
4.  Output:
5.  The predicted class label c for the input record r
6.  Start
7.  c = dagClassify(r, Root)
8.  End
9.  function dagClassify(r,Node)
10.     C = Classification result for r using Node classifier
11.     if |C| == 1 then
12.         return c (c ∈ C)
13.     else
14.         ChildNode = child node representing class group C
15.         return (dagclassify(r, ChildNode))
16.     End if
17. End function
```

The Non-rooted Directed Acyclic Graph (*non-rooted* DAG) Classification Model Framework

This section describes the proposed *non-rooted* DAG classification model. A simple example *non- rooted* DAG classifier for four class labels, C = {*a*, *b*, *c*, *d*}, is presented in Figure 3. The first level nodes are assigned class combinations of size three (|*C*| − 1), while the second level nodes are assigned class combinations of size two (|*C*| - 2). The distinction between this DAG structure and the rooted DAG structure can clearly be seen by comparison with Figure 2.

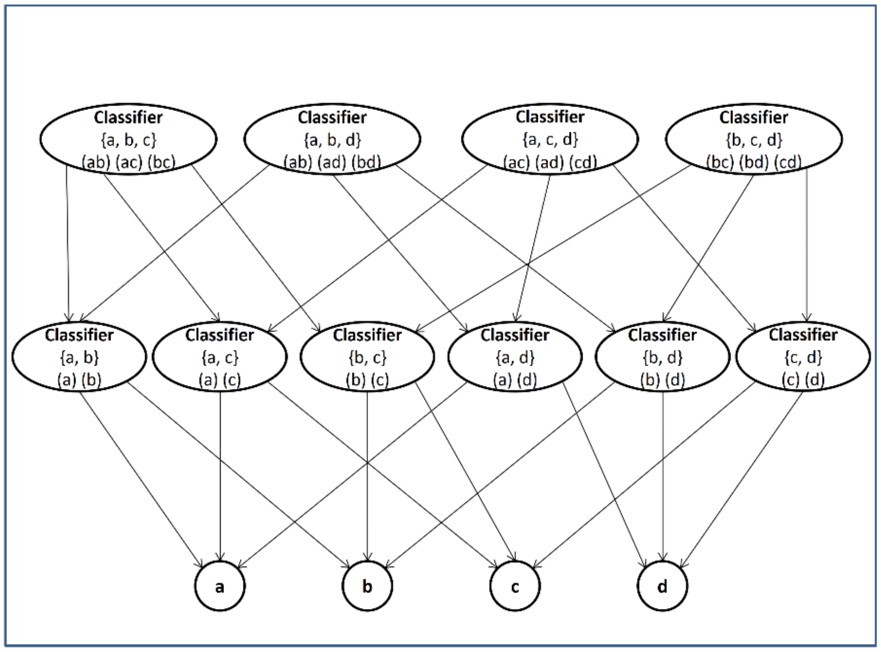

Figure 3. Non-rooted DAG example.

Non-rooted DAG Generation

The distinctions between the generation of the non- rooted DAG and rooted DAG are: (i) the elimination of the root node, and (ii) the breadth pruning applied during the generation process. Note that the breadth pruning is not shown in Figure 3 in which all nodes are generated except the root node. As mentioned in the introduction to this paper, breadth pruning cannot be applied in the case of the rooted DAG approach. This is because the rooted DAG requires the inclusion of all class combinations. More specifically we cannot create a root node, then eliminate nodes from the next level as this will result in a “null” references preventing the DAG classification from operating as intended. The aim of the breadth pruning is to eliminate *weak* classifiers that may exist at each DAG level, so that only strong classifiers are maintained as part of the proposed ensemble classification model. The potential advantages are: (i) improving the classification effectiveness by eliminating weak classifiers that can affect classification accuracy, and (ii) improving the complexity of the proposed model by reducing the number of nodes in the DAG model. The breadth pruning scheme is realized by utilizing the AUC (Area Under the receiver operating Curve) values generated when evaluating the internal classifiers, weak classifiers are then identified by their low associated AUC values. The breadth pruning process can be viewed as a two steps process: (i) pruning the first level nodes by evaluating them and pruning those nodes associated with low AUC values (based on a predefined AUC threshold) with the proviso that the classes associated with any node to be pruned are still covered by at least one remaining node in the level, and (ii) pruning the remaining levels by only generating nodes that are referred to by previous level nodes.

As noted earlier, the flexibility of the non-rooted DAG structure, and the adopted combination procedure, allows the generation of a DAG with any predefined number of levels. With respect to the work presented in this paper two different variations of the non-rooted DAG structure are considered: (i) the standard *non- rooted* DAG, and (ii) the *two-level* DAG. In the first variation the combination sizes range from $|C|$-1 to 2, while in the second the combination sizes range from 3 to 2 (only two levels generated). Note that breadth pruning is still applied in both cases. The conjecture here is that by reducing the number levels to 2 the classification performance (with respect to efficiency, effectiveness and scalability) of the DAG classification model will be enhanced, because: (i) the number of classifiers to be generated will be reduced, as a result the proposed model can be generated for datasets that feature larger number of class labels than would be possible otherwise; (ii) the internal classifiers are not required to discriminate between large numbers of class combinations; and (iii) the number of classifiers that are required to be evaluated during the classification stage will be decreased, as a result the probability of misclassification will also be decreased as well as the classification run time.

Non-rooted DAG Operation

In this section the operation of the suggested *non- rooted* DAG classification model is explained. Two methods of operation are considered: (i) the single path strategy and (ii) the multiple path strategy. A challenging issue associated with both strategies is how to identify the best *starting node* among the set of nodes at the first level in a given DAG. This is addressed by using the probability values associated with the Naive Bayes classifiers generated for each DAG node.

Single path strategy. The single path classification strategy can be viewed as a two-step process: (i) determine a best start node amongst the set of nodes available at the first level in the DAG by evaluating all the classifiers that exist at this level and selecting the node with the classifier that generates the highest probability value, and then (ii) drilling down as dictated by subsequent internal node classifications until a classifier that can assign a single class label to the given record is arrived at. Algorithm 3 presents the Single path procedure. The inputs to the algorithm are: (i) a new unseen record *r*; and (ii) a reference to the nodes at the first level in the given DAG, *FirstLevelNodes* (from which all the DAG child nodes can be identified). The output is a predicted class label for *r*. The process commences by identifying the best starting node among nodes at the first level in the DAG (line 8-13) by: (i) looping through the nodes at the first level (line 8), and (ii) for each node in the first level: classifying *r* using the respective node classifier (line 9), and adding the resulting class group, with the associated probability, to *S*, the set of class groups and associated probabilities resulting from evaluating first level nodes (line 10). The best start node is then the node with the highest associated probability value (line 12). The next node will be the child node for the identified *startNode* representing the class group associated with maximum probability value (line 13). The next step is a recursive process using the *dagclassify* function described in lines 17 to 25. The operation of the *dagclassify* function is the same as explained earlier for Algorithm 2.

Algorithm 3. Non-Rooted DAG single path classification

```
1.  Input
2.  r = A new unseen record
3.  FirstLevelNodes = nodes at DAG first level
4.  Output
5.  The predicted class label c for the input record r
6.  Start
7.   S = Results for r, using DAG first level classifiers, comprised of: (i) class groups
    and (ii) associated Bayesian probability values (initially S = {})
8.   for each Node in the FirstLevelNodes do
9.    classify r using Node classifier
10.   add the resulting class group with the associated Bayesian probability value to S
11.  Endfor
12.   startNode = node associated with the maximum probability value in S
13.  ChildNode = child node for startNode representing class group associated with
    the maximum probability
14.  c = dagclassify(r, ChildNode)
15. End
16.
17. function dagClassify(r, Node)
18.     C = Classification result for r using Node classifier
19.       if |C| == 1 then
20.          return c (c ∈ C)
21.     else
22.          ChildNode = child node representing class group C
23.         return (dagclassi f y(r, ChildNode))
24.     End if
25. End function
```

Multiple paths strategy. The multiple-path classification strategy can be viewed as a three-step process: (i) determine the start node(s) from the set of nodes available at the first level in the DAG by evaluating all the classifiers that exist at this first level and selecting one or two nodes as start nodes based on the probability threshold σ, (ii) for each identified node drill down following one or two paths as indicated and repeat until a classifier that can assign a single class label to the given record is arrived at, and finally (iii) identify the class label associated with the highest generated accumulated weight value. Algorithm 4 summarizes the multiple-path procedure. The inputs to the algorithm are: (i) the new unseen record *r*; (ii) a reference, *FirstLevelNodes*, to the first level DAG; and (iii) the path selection threshold σ. For simplicity the algorithm is decomposed into two main functions: *dagFirstLevelMultiPathClassify*, and *dagMultiPathClassify*.

Starting with the *dagFirstLevelMultiPathClassify* function, which is responsible for determining the start node (or nodes) amongst the set of nodes available at the first level. The process commences by evaluating all the classifiers that exist at the first level (lines 14-16), and selecting the two nodes that generate the highest probability values (lines 18-19). If the second highest probability value is greater than σ then both nodes will be considered as start nodes, otherwise only the node associated with the highest probability value will be considered as the start node (lines 18-21). After determining the start node(s) the recursive function *dagMultiPathClassify* is called. The *dagMultiPathClassify* function operates in a similar manner to the *dagClassify* function presented in Algorithm 3 except that it uses: (i) the σ threshold to decide whether one or two branches will be followed at each node; (ii) uses the variable *accumProb* to store the accumulated Bayesian probability values in a followed path; (iii) maintains a *counter* to count the number of probability values in a followed path; and (iv) uses a data structure, *Path*, in which to hold candidate class labels with their associated normalized Bayesian probability values. On each recursion of the *dagMultiPathClassify* function the Bayes classifier held at the current node is used to produce a probability value (lines 27-30) with respect to *r* for each class group. Only the class groups associated with the two highest probability values are considered (as maximum of two branches will be followed at each node). Whenever the size of a class group considered at a node is equal to one (lines 31 and 39), indicating that the group comprises a single class label, the class label and associated normalized probability value are added to *Path* (lines 33 and 41). Note that the normalized probability is calculated by dividing the accumulated probability generated so far, *accumProb*, by the number of classifiers used in the current path, *counter* (lines 32 and 40). Whether one or two paths are followed depends on the probability values returned using the Bayes classifier at the current node and the σ threshold. If the second highest probability value is greater than σ (line 38) then two paths will be followed, otherwise only a single path will be followed. At the end of the process the *Path* data structure is processed to identify the class label with the highest associated normalized probability value (line 8).

Experiments and Evaluation

This section presents an overview of the adopted experimental set up and an evaluation of the results obtained. Twelve datasets, with various numbers of class labels, were used to evaluate the effectiveness of the proposed DAG classification approaches. These datasets were taken from the UCI data repository (Bache & Lichman, 2013), and were pre-processed using LUCS-KDD-DN software (Coenen, 2003). Ten-fold Cross Validation (TCV) was used throughout. The evaluation measures used were average accuracy and average AUC. To determine whether the results obtained were statistically significant, or not, the Wilcoxon signed rank test, for comparing two classification models, and the Friedman test (for comparing several classification models) were used. All experiments were conducted using a 2.7 GHz Intel Core i5 with 16 GB 1333 MHz DDR3 memory, running OS X 10.9.2 (13C64). In addition to the suggested DAG classification approaches (*rooted* DAG, *non-rooted* DAG, and *two-level* DAG), the datasets were also classified using: (i) a Binary Tree hierarchical classification model, (ii) a stand-alone Naive Bayes classifier, and (iii) a Bagging classifier.

The objectives of the evaluation were as follows:

1. To compare the operation of the suggested DAG approaches, *rooted* DAG, *non-rooted* DAG, and *two-level* DAG
2. To compare the use of the single and multiple path strategies with respect to each of the DAG approaches.
3. To compare the operation of the proposed DAG hierarchical ensemble classification with a simple binary tree hierarchical ensemble model.

Algorithm 4. Non-Rooted DAG Multi-Path

```
1.  **Input:**
2.  r = A new unseen record
3.  FirstLevelNodes = nodes at the first level in the DAG
4.  $\sigma$ = Path selection threshold
5.  **Output**
6.  The predicted class label c for the input record r
7.  **Start**
8.  Path = Set of identified paths each comprised of: (i) a class label and (ii) an associated
    normalised Bayesian probability value, initially Path = {}
9.   dagFirstLevelMultiPathClassify(r, FirstLevelNodes)
10.  c = Class label with highest probability value in Path
11.  **End**
12. **function** dagFirstLevelMultiPathClassify(r, FirstLevelNodes)
13.   S = Classification results for r using the classifiers at the first level in the DAG
    comprised of: (i) class groups, and (ii) the associated Bayesian probability values (initially
    S = {})
14.   **for** each Node in the FirstLevelNodes **do**
15.   classify r using Node classifier
16.   add the resulting class group with the associated Bayesian probability value to S
17.  **Endfor**
18.    startNode1 = node associated with the highest probability value in S
19.    startNode2 = node associated with the second highest probability value in S
20.    dagMultiPathClassify(r, startNode1, 0, 0)
21.    **if** second highest probability in $S \geq \sigma$ **then**
22.        dagMultiPathClassify(r, startNode2, 0, 0)
23.    **end if**
24. **end function**
25. **function** dagMultiPathClassify(r, Node,accumProb,counter)
26.   classify r using Node classifier
27.    $C_1$ = Class group in C associated with highest probability value
28.    $p_1$ = Bayesian probability associated with $C_1$
29.   $C_2$ = Class group in C associated with second highest probability value
30.    $p_2$ = Bayesian probability associated with $C_2$
31.    **if** $|C_1| == 1$ **then**
32.        normProb = (AccumProb + $p_1$)/(counter + 1)
33.       Path = Path $\cup \langle$ c, normProb$\rangle$  ($c \in C_1$ )
34.    **else**
35.      ChildNode = child node representing class group $C_1$
36.       dagMultiPathClassify(r,ChildNode,accumProb+$p_1$,counter+1)
37.    **end if**
38.    **if** $p_2 \geq \sigma$ **then**
39.       **if** $|C_2| == 1$ **then**
40.           normProb = (AccumProb + $p_2$)/(countert + 1)
41.          Path = Path $\cup \langle$ c, normProb$\rangle$  ($c \in C_2$ )
42.       **else**
43.          ChildNode = child node representing class group $C_2$
44.             dagMultiPathClassify(r,ChildNode,accumProb+$p_2$,counter+1)
45.         **end if**
46.     **end if**
47. **end function**
```

4. To compare the operation of the proposed DAG hierarchical ensemble classification with stand-alone classification and with an alternative well-known ensemble method (bagging).
5. To compare the run time, for both the training and testing stages, for all the methods considered.

The results in the context of the above evaluation objectives are discussed in the following sub-sections.

Comparison Between DAG Based Hierarchical Classification Approaches

This section presents a comparison between the operation of the three DAG variations: (i) *rooted* DAG, (ii) *non-rooted* DAG, and (iii) *two-level* DAG, each coupled with either the single or the multiple path strategy. The objective was to determine the most effective and efficient DAG structure and to compare the use of the single and multiple path strategies for hierarchical ensemble classification as a solution for successive misclassification. With respect to the Multiple Path strategy a threshold of $\sigma = 0.7 \times 10^{-4}$ was used with the rooted DAG, and $\sigma = 0.1 \times 10^{-4}$ with the non-rooted and two-level DAG. Experiments using a range of alternative σ values (not reported here because of space limitations) were conducted from which it was concluded that $\sigma = 0.7 \times 10^{-4}$ and $\sigma = 0.1 \times 10^{-4}$ were the most appropriate thresholds. Regarding the AUC threshold value used in the breadth pruning with respect to the non-rooted and two-level DAG approaches, experiments using a range of alternative AUC values were conducted from which it was demonstrated that identifying a different threshold value for each dataset is better than identifying a single threshold value for all datasets because this value will affect the multiple-path results if it is not the most appropriate value for the specific dataset.

The obtained results, using the DAG approaches and with respect to the ten datasets considered, are presented in Tables 1 and 2. For simplicity, and because the evaluation datasets include unbalanced datasets, the results will be discussed according to the AUC values presented in Table 2. From this table it can firstly be observed that following multiple paths within the DAG classification model (especially when using rooted and non-rooted DAGs) improves the classification performance. Comparing the operation of the three approaches it can be clearly seen that the two-level DAG, which combines depth and breadth pruning, outperformed the rooted and non-rooted DAG approaches. More specifically, the average (mean) AUC obtained from using the two-level DAG approach for the ten datasets was 0.63, while rooted DAG and non-rooted DAG produced average AUC results of 0.62, when multiple paths were followed on both cases. Although there is a noticeable differences in the effectiveness of the considered DAG variations, at least according to average recorded AUC values, these differences were not found to be statistically significant according to the conducted Friedman test.

The results obtained for the run-time experiments with respect to the DAG based approaches are presented in Table 3. The table presents the generation and classification time for each DAG approach. From the table it can be observed that the *two-level* DAG structure requires the least generation time, as well as, the least classification time when the multiple path strategy is adopted. It is also clear that the multiple paths strategy consumes more time than the single path strategy for all the DAG variations.

Table 1. Average Accuracy values obtained using the DAG based classification approaches.

Dataset	**Classes**	**Single Path Strategy**			**Multiple Path Strategy**		
		Rooted	**Non-Rooted**	**Two-Level**	**Rooted**	**Non-Rooted**	**Two-Level**
Nursery	5	90.26	79.83	**91.44**	90.28	82.32	90.02
Heart	5	55.91	57.01	**59.91**	55.37	56.25	59.64
PageBlocks	5	**92.69**	91.83	92.02	**92.65**	91.87	92.05
Dermatology	6	**87.23**	**87.23**	86.09	**87.23**	**87.23**	85.51
Glass	7	69.81	69.81	57.58	**72.99**	71.16	57.18
Zoo	7	**92.18**	**92.18**	**92.18**	**92.18**	**92.18**	**92.18**
Ecoli	8	**84.43**	**84.43**	82.40	82.56	82.26	80.89
Led	10	75.66	75.66	**75.75**	75.56	75.53	75.66
PenDigits	10	83.58	83.59	**83.84**	83.58	83.59	**83.84**
Soybean	15	**90.75**	90.57	90.04	**90.75**	90.57	90.04
Mean		82.25	81.21	81.23	**82.32**	81.30	80.80

Table 2. Average AUC values obtained using the DAG based classification approaches.

Dataset	**Single Path Strategy**			**Multiple Path Strategy**		
	Rooted	**Non-Rooted**	**Two-Level**	**Rooted**	**Non-Rooted**	**Two-Level**
Nursery	0.45	0.40	0.54	0.45	0.41	**0.58**
Heart	0.35	0.39	**0.40**	0.35	0.37	**0.40**
PageBlocks	0.52	0.53	0.49	0.52	**0.54**	0.47
Dermatology	0.85	**0.85**	0.84	**0.85**	**0.85**	0.84
Glass	0.46	0.46	0.48	**0.51**	0.50	0.49
Zoo	0.58	0.58	**0.59**	0.58	**0.59**	**0.59**
Ecoli	**0.41**	**0.41**	0.40	0.38	0.38	0.39
Led	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**
PenDigits	0.83	**0.84**	**0.84**	0.83	**0.84**	**0.84**
Soybean	**0.92**	**0.92**	**0.92**	**0.92**	**0.92**	**0.92**
Mean	0.61	0.61	**0.63**	0.62	0.62	**0.63**

Table 3: Run time results (in seconds) obtained using DAG based classification approaches.

Dataset	**Generation Time**			**Single Path Classification Time**			**Multiple Path Classification Time**		
	Rooted	**Non-Rooted**	**Two-Level**	**Rooted**	**Non-Rooted**	**Two-Level**	**Rooted**	**Non-Rooted**	**Two-Level**
Nursery	5.982	4.380	3.142	0.012	0.007	0.010	0.595	0.601	0.625
Heart	0.333	0.299	0.245	0.001	0.001	0.001	0.015	0.017	0.016
PageBlocks	2.510	2.043	1.408	0.008	0.004	0.003	0.266	0.274	0.261
Dermatology	0.445	0.377	0.288	0.001	0.001	0.001	0.020	0.021	0.019
Glass	0.539	0.415	0.261	0.001	0.000	0.001	0.016	0.017	0.013
Zoo	0.491	0.360	0.221	0.001	0.000	0.001	0.009	0.009	0.007
Ecoli	1.032	0.801	0.342	0.001	0.000	0.001	0.031	0.034	0.019
Led	33.701	22.142	1.097	0.011	0.005	0.022	0.261	0.266	0.163
PenDigits	264.369	150.180	4.508	0.039	0.033	0.047	0.723	0.658	0.526
Soybean	1520.706	639.260	1.467	0.009	0.003	0.025	0.068	0.063	0.057
Mean	183.011	82.026	**1.298**	0.008	**0.005**	0.011	0.200	0.196	**0.171**

Comparison Between DAG Based Hierarchical Classification and Binary Tree Based Hierarchical Classification

This section presents a comparison between a binary tree hierarchical ensemble model, and the *two-level* DAG classification approach (the previous section has established that the *two-level* DAG structure produces the best classification performance). With respect to the binary tree classifier, a Naive Bayes classifier was generated for each tree node, and data segmentation was used to distribute class labels between nodes within the tree, both single path and multi-path strategies were considered. Because of the efficiency of the *two-level* DAG approach, compared to the *rooted* and *non-rooted* DAG approaches, in this section we include results obtained using two further datasets that featured large numbers of class labels (Chess KRvK, and Letter Recognition) in addition to the ten datasets presented in Table 1 and 2. Table 4 shows the results obtained (best results highlighted in bold font). From the table it can be clearly observed that the *two-level* DAG approach (using either the single and the multiple path strategies) outperformed the binary tree hierarchical ensemble classification model for most of the datasets considered in the evaluation, especially datasets that featured large numbers of class labels such as: Led, Pen Digits, Soybean, and Letter Recognition (notice the last column in the table). According to the conducted statistical tests, usage of the DAG structure was found to be significantly more effective with respect to the generation of the hierarchical classification model than the Binary Tree structure, regardless of the adopted classification strategy (Single or Multiple Path). Unfortunately space limitations preclude the presentation of a detailed analysis of the run time results; however, it can be seen that the binary tree approach clearly requires less run time (because the proposed DAG structure is more complex).

Comparison Between Stand-Alone Classification, Bagging, and DAG Based Hierarchical Classification

This section presents the results obtained from a comparison between the operation of: a "conventional" form of classification using a single Naive Bayes classifier, a Bagging ensemble classifier and the two-level DAG approach. The results are presented in Table 5. From the table it can be observed that the best average AUC value, with respect to the twelve datasets considered, was obtained when using the two-level DAG approach. It is interesting to note that the proposed two-level DAG tends to improve the classification effectiveness for unbalanced datasets such as: Nursery, Heart, PageBlocks, Glass, Ecoli and chess KRvK. It is conjectured that the combination techniques, used to distribute class labels between nodes within the DAG, helps in the handling of unbalanced datasets. More specifically, instead of letting a single classifier handle an unbalanced dataset, the combination mechanism distributes classes between DAG nodes, some nodes will handle unbalanced subsets while other nodes will handle balanced subsets. During the classification stage only a few good quality classifiers will then be used to predict the class label for a given record, there is thus opportunity for the classifiers used to operate using balanced subsets. Consequently it is conjectured that good results are likely to be obtained. With respect to the statistical evaluation, it was found that there was no statistically significant difference in effectiveness between Naive Bayes classification, Bagging of Naive Bayes classifiers and Naive Bayes DAG.

Table 4. Average Accuracy and AUC values obtained using a Binary Tree classification model and the proposed *two-level* DAG approach.

Dataset	**Single Path**				**Multiple Path**			
	Binary Tree		**Two-level DAG**		**Binary Tree**		**Two-level DAG**	
	ACC.	**AUC**	**ACC.**	**AUC**	**ACC.**	**AUC**	**ACC.**	**AUC**
Nursery	90.12	0.44	**91.44**	0.54	89.09	**0.58**	90.02	**0.58**
Heart	57.70	**0.41**	**59.91**	0.40	53.77	0.36	59.64	0.40
PageBlocks	91.96	0.34	92.02	**0.49**	91.27	0.48	**92.05**	0.47
Dermatology	79.80	0.79	86.09	**0.84**	**84.60**	**0.84**	85.51	**0.84**
Glass	**63.94**	0.43	57.58	0.48	55.28	**0.51**	57.18	0.49
Zoo	**93.18**	**0.59**	**93.18**	**0.59**	92.18	0.58	**93.18**	**0.59**
Ecoli	82.31	0.36	**82.40**	**0.40**	64.15	0.27	80.89	0.39
Led	60.16	0.60	**75.75**	**0.76**	61.13	0.61	75.66	**0.76**
PenDigits	68.56	0.68	**83.84**	**0.84**	81.18	0.81	**83.84**	**0.84**
Soybean	79.55	0.81	**90.04**	**0.92**	83.71	0.83	**90.04**	**0.92**
ChessKRvK	35.18	0.27	34.58	0.33	33.88	**0.37**	**35.36**	0.36
LetterRecog.	39.16	0.39	**55.85**	**0.56**	53.44	0.53	55.84	**0.56**
Mean	70.14	0.51	**75.22**	**0.60**	70.31	0.56	74.93	**0.60**

Table 5. Average Accuracy and AUC values obtained using stand-alone Naive Bayes classification, Bagging and the proposed *two-level* DAG classification approach.

Dataset	**Naïve Bayes**		**Bagging Ensemble**		**Two-level DAG**	
	ACC.	**AUC**	**ACC.**	**AUC**	**ACC.**	**AUC**
Nursery	**90.22**	0.45	89.96	0.46	90.02	**0.58**
Heart	54.60	0.34	51.28	0.30	**59.91**	**0.40**
PageBlocks	**92.69**	**0.52**	92.62	**0.52**	92.02	0.49
Dermatology	**86.66**	**0.85**	81.00	0.81	86.09	0.84
Glass	**67.83**	**0.49**	55.28	0.46	57.18	**0.49**
Zoo	92.27	0.59	**94.27**	**0.62**	93.18	0.59
Ecoli	81.70	0.38	82.56	0.39	**82.40**	**0.40**
Led	75.59	**0.76**	75.50	**0.76**	**75.75**	**0.76**
PenDigits	**84.94**	**0.85**	84.57	**0.85**	83.84	0.84
Soybean	**91.11**	**0.93**	86.83	0.89	90.04	0.92
ChessKRvK	**36.32**	0.33	35.66	0.34	35.36	**0.36**
LetterRecog.	**57.37**	**0.57**	56.93	**0.57**	55.85	0.56
Mean	**75.94**	0.59	73.87	0.58	75.14	**0.60**

The results obtained for the run-time experiments with respect to the conventional Naive Bayes classification and the Bagging ensemble (and the two-level DAG) are presented in Table 6. From the table it can be observed that the lowest generation and classification time was recorded when using the single Naive Bayes classifier. However, although the two-level DAG takes longer to be generated, the model needs only to be generated once after which it can be used repeatedly.

Table 6. Run time results (in seconds) obtained using stand-alone Naive Bayes classification, Bagging and the proposed *two-level* DAG classification approach.

Dataset	**Naïve Bayes**		**Bagging**		**Two-level DAG**	
	Gen.Time	**Class.Time**	**Gen.Time**	**Class.Time**	**Gen.Time**	**Class.Time**
Nursery	0.974	0.003	1.180	0.011	3.142	0.625
Heart	0.202	0.000	0.216	0.001	0.245	0.016
PageBlocks	0.676	0.001	0.775	0.005	1.408	0.261
Dermatology	0.242	0.000	0.296	0.000	0.288	0.019
Glass	0.178	0.000	0.182	0.000	0.261	0.013
Zoo	0.163	0.000	0.136	0.001	0.221	0.007
Ecoli	0.206	0.000	0.208	0.000	0.342	0.019
Led	0.529	0.002	0.547	0.004	1.097	0.163
PenDigits	1.100	0.006	1.121	0.010	4.508	0.526
Soybean	0.353	0.001	0.329	0.003	1.467	0.057
ChessKRvK	1.470	0.006	1.674	0.008	70.401	1.881
LetterRecog.	1.398	0.007	1.580	0.011	76.011	4.321
Mean	**0.624**	**0.002**	0.687	0.005	13.283	0.659

Conclusion and Future Work

In this paper a DAG hierarchical ensemble classification model has been presented as a solution to the multi-class classification problem. Broadly the approach entails: generating a Naive Bayes classifier for each DAG node and the use of combination techniques to distribute class labels between nodes within the DAG. Three DAG variations were proposed: (i) *rooted* DAG, (ii) *non-rooted* DAG and (iii) *two-level* DAG. In addition a strategy for following multiple paths within the DAG model was proposed as a solution to the successive misclassification issue associated with hierarchical classification. From the reported experimental results it was demonstrated that there was no statistically significant difference in effectiveness between the different proposed DAG variations. However, the *two-level* DAG, with depth and breadth pruning, is the most efficient. Of course, scalability is another advantage of the *two-level* DAG where the DAG classification model can be generated for data sets that feature larger numbers of class labels, such as the Chess KRvK and Letter Recognition data sets used as part of the evaluation. According to the statistical tests results, no statistically significant difference in performance between single path and multiple path strategies, regardless of the adopted DAG variation, was identified. The reason for this is that the combination technique used to distribute classes between nodes in the DAG resulted in well-defined class labels at each DAG node; consequently the number of misclassifications is less and the effect of following multiple paths within the DAG is not highly significant. With respect to the comparison with the binary tree ensemble classification model, the results showed that the *two-level* DAG significantly outperforms the binary tree ensemble classification model; the suggested reason for this is that the misclassification issue is handled well by the combination mechanism and the pruning techniques used, in addition to the multiple path strategy. The evaluation also indicated that the proposed *two-level* DAG hierarchical classification approach could be successfully used to classify data in a more effective manner than when stand-alone classifiers (or types of other ensemble classifier such as bagging) were used in the context of some data sets considered in the evaluation, especially unbalanced datasets. For additional improvements the authors intend to investigate further techniques whereby depth and breadth pruning can be applied so as to reduce the overall numbers of classifiers within the DAG structures, it is consequently conjectured that an even higher classification accuracy might be obtained.

References

Alshdaifat, E., Coenen, F., Dures, K. (2013). Hierarchical single label classification: An alternative approach. In Max Bramer and Miltos Petridis (Eds.)*, the thirty-third BCS SGAI International Conference on Artificial Intelligence* (pp. 39–52).

Alshdaifat, E., Coenen, F., Dures, K. (2013). Hierarchical classification for solving multi-class problems: A new approach using naive bayesian classification. In H. Motoda, Z. Wu, L. Cao, O. R. Za¨iane, M. Yao, W. Wang (Eds.), *the 9th International Conference on Advanced Data Mining and Applications* (pp. 493–504).

Alshdaifat, E., Coenen, F., Dures, K. (2014). A multi-path strategy for hierarchical ensemble classification. In P. Perner (Ed.), *Machine Learning and Data Mining in Pattern Recognition* (pp. 198– 212).

Athimethphat, M., Lerteerawong, B. (2012). Binary classification tree for multiclass classification with observation-based clustering. *In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)* (pp. 1–4).

Bache, K., Lichman, M. (2013). UCI machine learning repository. From http://archive.ics.uci.edu/ml.

Bauer, E., Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, *Machine learning, 36*, 105–139.

Beygelzimer, A., Langford, J., Ravikumar, P. (2007). Multiclass Classification with Filter Trees. Retrieved Jun 12, 2013, from http://mi.eng.cam.ac.uk/~mjfg/local/Projects/filter_tree.pdf

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24 (2),* 123–140.

Breiman, L. (2001). Random forests. *Machine Learning, 45,* 5–32.

Chen, Y., Crawford, M.M., Ghosh, J. (2004). Integrating support vector machines in a hierarchical output decomposition framework. *In Proceedings of the 2004 IEEE International Geoscience and Remote Sensing Symposium* (pp. 949–953).

Coenen, F. (2003). The LUCS-KDD discretised/normalised arm and carm data library. From http://www.csc.liv.ac.uk/ frans/KDD/Software /LUCS KDD DN.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems* (pp. 1–15). London, UK.

Freund, Y., Schapire, R. Abe, N. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence 14 (5),* 771–80.

Hansen, L. K., Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (10),* 993-1001.

Jiawei, H., Micheline, K., & Jian, P. (2011). *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann.

Kumar, S., Ghosh, J., Crawford, M. (2002). Hierarchical fusion of multiple classifiers for hyperspectral data analysis, *Pattern Analysis and Applications, 5 (2),* 210–220.

Lei, H., Govindaraju, V. (2005). Half-against-half multi-class support vector machines. *In Proceedings of the 6th International Workshop on Multiple Classifier Systems* (pp 156-164). Seaside, CA, USA.

Machov, K., Bark, F., Bednr, Bednar, P. (2006). A bagging method using decision trees in the role of base classifiers. *Acta Polytechnica Hungarica, 3 (2),* 121-132.

Madzarov, G., Gjorgjevikj, D., Chorbev, I. (2009). A multi-class svm classifier utilizing binary decision tree. *Informatica, 33 (2),* 233-241.

Opitz, D., Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research, 11*, 169-198.

Oza, N., Tumer, K. (2008). Classifier ensembles: Select real-world applications. *Information Fusion, 9 (1),* 4–20.

Quinlan, J. R. (1996). Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press* (pp. 725–730).

Songsiri, P., Phetkaew, T., Kijsirikul B. (2015). Enhancements of multi-class support vector machine construction from binary learners using generalization performance, *Neurocomputing*, 151 (1), 434–448.

Wirth, J., Catlett, J. (1988). Experiments on the costs and benefits of windowing in id3. *In J. E. Laird (Ed.), ML,* Morgan Kaufmann (pp. 87–99).

Zhou, Z. (2009). Ensemble learning. *In: S. Z. Li, A. K. Jain (Eds.). Encyclopedia of Biometrics, Springer US* (pp. 270–273).