Chapter 40
RBDT-1 Method: Combining Rules and Decision Tree Capabilities

Amany Abdelhalim and Issa Traore

Abstract Most of the methods that generate decision trees for a specific problem use examples of data instances in the decision tree generation process. This chapter proposes a method called “*RBDT-1*” - rule based decision tree - for learning a decision tree from a set of decision rules that cover the data instances rather than from the data instances themselves. *RBDT-1* method uses a set of declarative rules as an input for generating a decision tree. The method’s goal is to create on-demand a short and accurate decision tree from a stable or dynamically changing set of rules. We conduct a comparative study of *RBDT-1* with existing decision tree methods based on different problems. The outcome of the study shows that in terms of tree complexity (number of nodes and leaves in the decision tree) *RBDT-1* compares favorably to *AQDT-1, AQDT-2* which are methods that create decision trees from rules. *RBDT-1* compares favorably also to *ID3* which is a famous method that generates decision trees from data examples. Experiments show that the classification accuracies of the different decision trees produced by the different methods under comparison are equal.

1 Introduction

Decision Tree is one of the most popular classification algorithms used in data mining and machine learning to create knowledge structures that guide the decision making process.

The most common methods for creating decision trees are those that create decision trees from a set of examples (data records). We refer to these methods as data-based decision tree methods.

A. Abdelhalim (✉)
Department of Electrical and Computer Engineering, University of Victoria, 3055 STN CSC, Victoria, B.C.,V8W 3P6, Canada
e-mail: amany@ece.uvic.ca

On the other hand, to our knowledge there are only few approaches that create decision trees from rules proposed in the literature which we refer to as rule-based decision tree methods.

A decision tree can be an effective tool for guiding a decision process as long as no changes occur in the dataset used to create the decision tree. Thus, for the data-based decision tree methods once there is a significant change in the data, restructuring the decision tree becomes a desirable task. However, it is difficult to manipulate or restructure decision trees. This is because a decision tree is a procedural knowledge representation, which imposes an evaluation order on the attributes. In contrast, rule-based decision tree methods handle manipulations in the data through the rules induced from the data not the decision tree itself. A declarative representation, such as a set of decision rules is much easier to modify and adapt to different situations than a procedural one. This easiness is due to the absence of constraints on the order of evaluating the rules [1].

On the other hand, in order to be able to make a decision for some situation we need to decide the best order in which tests should be evaluated in those rules. In that case a decision structure (e.g. decision tree) will be created from the rules. So, rule-based decision tree methods combine the best of both worlds. On one hand they easily allow changes to the data (when needed) by modifying the rules rather than the decision tree itself. On the other hand they take advantage of the structure of the decision tree to organize the rules in a concise and efficient way required to take the best decision. So knowledge can be stored in a declarative rule form and then be transformed (on the fly) into a decision tree only when needed for a decision making situation [1].

In addition to that, generating a decision structure from decision rules can potentially be performed faster than generating it from training examples because the number of decision rules per decision class is usually much smaller than the number of training examples per class. Thus, this process could be done on demand without any noticeable delay [2, 3]. Data-based decision tree methods require examining the complete tree to extract information about any single classification. Otherwise, with rule-based decision tree methods, extracting information about any single classification can be done directly from the declarative rules themselves [4].

Although rule-based decision tree methods create decision trees from rules, they could be used also to create decision trees from examples by considering each example as a rule. Data-based decision tree methods create decision trees from data only. Thus, to generate a decision tree for problems where rules are provided, e.g. by an expert, and no data is available, rule-based decision tree methods are the only applicable solution.

This chapter presents a new rule-based decision tree method called *RBDT-1*. To generate a decision tree, the *RBDT-1* method uses in sequence three different criteria to determine the fit (best) attribute for each node of the tree, referred to as *the attribute effectiveness (AE)*, *the attribute autonomy (AA)*, and *the minimum value distribution (MVD)*. In this chapter, the *RBDT-1* method is compared to the *AQDT-1* and *AQDT-2* methods which are rule-based decision tree methods, along with *ID3* which is one of the most famous data-based decision tree methods.

2 Related Work

There are few published works on creating decision structures from declarative rules.

The *AQDT-1* method introduced in [1] is the first approach proposed in the literature to create a decision tree from decision rules. The *AQDT-1* method uses four criteria to select the fit attribute that will be placed at each node of the tree. Those criteria are the *cost*[1], the *disjointness*, the *dominance*, and the *extent*, which are applied in the same specified order in the method's default setting.

The *AQDT-2* method introduced in [4] is a variant of *AQDT-1*. *AQDT-2* uses five criteria in selecting the fit attribute for each node of the tree. Those criteria are the *cost*[1]*, disjointness, information importance, value distribution*, and *dominance*, which are applied in the same specified order in the method's default setting. In both the *AQDT-1 & 2* methods, the order of each criterion expresses its level of importance in deciding which attribute will be selected for a node in the decision tree. Another point is that the calculation of the second criterion – the *information importance* – in *AQDT-2* method depends on the training examples as well as the rules, which contradicts the method's fundamental idea of being a rule-based decision tree method. *AQDT-2* requires both the examples and the rules to calculate the *information importance* at certain nodes where the first criterion – *Disjointness* – is not enough in choosing the fit attribute. *AQDT-2* being both dependent on the examples as well as the rules increases the running time of the algorithm remarkably in large datasets especially those with large number of attributes.

In contrast the *RBDT-1* method, proposed in this work, depends only on the rules induced from the examples, and does not require the presence of the examples themselves.

Akiba et al. [5] proposed a rule-based decision tree method for learning a single decision tree that approximates the classification decision of a majority voting classifier. In their proposed method, if-then rules are generated from each classifier (a *C4.5* based decision tree) and then a single decision tree is learned from these rules. Since the final learning result is represented as a single decision tree, problems of intelligibility and classification speed and storage consumption are improved. The procedure that they follow in selecting the best attribute at each node of the tree is based on the *C4.5* method which is a data-based decision tree method. The input to the proposed method requires both the real examples used to create the

[1]In the default setting, the *cost* equals 1 for all the attributes. Thus, the *disjointness* criterion is treated as the first criterion of the *AQDT-1* and *AQDT-2* methods in the decision tree building experiments throughout this paper.

classifiers (decision trees) and the rules extracted from the classifiers, which are used to create a set of training examples to be used in the method.

In [6], the authors proposed a method called Associative Classification Tree (ACT) for building a decision tree from association rules rather than from data. They proposed two splitting algorithms for choosing attributes in the *ACT* one based on the confidence gain criterion and one based on the entropy gain criterion. The attribute selection process at each node in both splitting algorithms relies on both the existence of rules and the data itself as well. Unlike our proposed method *RBDT-1, ACT* is not capable of building a decision tree from the rules in the absence of data, or from data (considering them as rules) in the absence of rules.

3 Rule Generation and Notations

In this section, we present the notations used to describe the rules used by our method. We also present the methods used to generate the rules that will serve as input to the rule-based decision tree methods in our experiments.

3.1 Notations

In order for our proposed method to be capable of generating a decision tree for a given dataset, it has to be presented with a set of rules that cover the dataset. The rules will be used as input to *RBDT-1* which will produce a decision tree as an output. The rules can either be provided up front, for instance, by an expert or can be generated algorithmically.

Let $a_1, ..., a_n$ denote the attributes characterizing the data under consideration, and let $D_1, ..., D_n$ denote the corresponding domains, respectively (i.e. D_i represents the set of values for attribute a_i). Let $c_1, ..., c_m$ represent the decision classes associated with the dataset.

3.2 Rule Generation Method

The *RBDT-1* takes rules as input and produces a decision tree as output. Thus, to illustrate our approach and compare it to existing similar approaches, we use in this chapter two options for generating a set of disjoint rules for each dataset. The first option is based on extracting rules from a decision tree generated by the *ID3* method, where we convert each branch – from the root to a leaf – of the decision tree to an if–then rule whose condition part is a pure conjunction. This scenario will ensure that we will have a collection of disjoint rules. We refer to these rules as *ID3-based* rules.

Using *ID3-based* rules will give us an opportunity to illustrate our method's capability of producing a smaller tree while using the same rules extracted from a decision tree generated by a data-based decision tree method without reducing the tree's accuracy.

The second rule generation option consists of using an *AQ-type* rule induction program. We use the *AQ19* program for creating logically disjoint rules which we refer to as *AQ-based* rules.

4 RBDT-1 Method

In this section, we describe the *RBDT-1* method by first outlining the format of the input rules used by the method and the attribute selection criteria of the method. Then we summarize the main steps of the underlying decision tree building process and present the technique used to prune the rules.

4.1 Attribute Selection Criteria

The *RBDT-1* method applies three criteria on the attributes to select the fittest attribute that will be assigned to each node of the decision tree. These criteria are *the Attribute Effectiveness*, *the Attribute Autonomy*, *and the Minimum Value Distribution.*

4.1.1 Attribute Effectiveness (AE)

AE is the first criterion to be examined for the attributes. It prefers an attribute which has the most influence in determining the decision classes. In other words, it prefers the attribute that has the least number of "don't care" values for the class decisions in the rules, as this indicates its high relevance for discriminating among rule sets of given decision classes. On the other hand, an attribute which is omitted from all the rules (i.e. has a "don't care" value) for a certain class decision does not contribute in producing that corresponding decision. Choosing attributes based on this criterion maximizes the chances of reaching leaf nodes faster which on its turn minimizes the branching process and leads to producing a smaller tree.

Using the notation provided above (see Section 3), letV_{ij} denote the set of values for attribute a_j involved in the rules in R_i, which denote the set of rules associated with decision class c_i, $1 \leq i \leq m$. Let DC denote the 'don't care' value, we calculate $C_{ij}(DC)$ as shown in (1):

$$C_{ij}(DC) = \begin{cases} 1 & \textit{if } DC \in V_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Given an attribute a_j, where $1 \leq j \leq n$, the corresponding attribute effectiveness is given in (2).

$$AE(a_j) = \frac{m - \sum\limits_{i=1}^{m} C_{ij}(DC)}{m} \tag{2}$$

(where m is the total number of different classes in the set of rules).

The attribute with the highest *AE* is selected as the fit attribute.

4.1.2 Attribute Autonomy (AA)

AA is the second criterion to be examined for the attributes. This criterion is examined when the highest *AE* score is obtained by more than one attribute. This criterion prefers the attribute that will decrease the number of subsequent nodes required ahead in the branch before reaching a leaf node. Thus, it selects the attribute that is less dependent on the other attributes in deciding on the decision classes. We calculate the attribute autonomy for each attribute and the one with the highest score will be selected as the fit attribute.

For the sake of simplicity, let us assume that the set of attributes that achieved the highest *AE* score are $a_1, ..., a_s, 2 \leq s \leq$ n. Let $v_{j1}, ..., v_{jp_j}$ denote the set of possible values for attribute a_j including the "don't care", and R_{ji} denote the rule subset consisting of the rules that have a_j appearing with the value v_{ji}, where $1 \leq j \leq s$ and $1 \leq i \leq p_j$. Note that R_{ji} will include the rules that have don't care values for a_j as well.

The *AA* criterion is computed in terms of the Attribute Disjointness Score (ADS), which was introduced by Imam and Michalski [1]. For each rule subset R_{ji}, let $MaxADS_{ji}$ denote the maximum ADS value and let ADS_List_{ji} denote a list that contains the *ADS* score for each attribute a_k, where $1 \leq k \leq s, k \neq j$.

According to [1], given an attribute a_j and two decision classes c_i and c_k (where $1 \leq i, k \leq m; 1 \leq j \leq s$), the degree of disjointness between the rule set for c_i and the rule set for c_j with respect to attribute a_j is defined as shown in (3):

$$ADS(A_j, C_i, C_k) = \begin{cases} 0 & if\, V_{ij} \subseteq V_{kj} \\ 1 & if\, V_{ij} \supseteq V_{kj} \\ 2 & if\, V_{ij} \cap V_{kj} \neq (\varnothing \text{ or } V_{ij} \text{ or } V_{kj}) \\ 3 & if\, V_{ij} \cap V_{kj} = \varnothing \end{cases} \tag{3}$$

The *Attribute Disjointness* of the attribute a_j, $ADS(a_j)$ score, is the summation of the degrees of class disjointness $ADS(a_j, c_i, c_k)$ given in (4):

$$ADS(a_j) = \sum_{i=1}^{m} \sum_{\substack{1 \leq k \leq s \\ i \neq k}} ADS(a_j, c_i, c_k) \tag{4}$$

Thus, the number of *ADS_List* that will be created for each attribute a_j as well as the number of *MaxADS* values that are calculated will be equal to p_j. The $MaxADS_{ji}$ value as defined by Wojtusiak [8] is $3 \times m \times (m-1)$ where m is the total number of classes in R_{ji}. We introduce the *AA* as a new criterion for attribute a_j as given in (5):

$$AA(a_j) = \frac{1}{\sum_{i=1}^{p_j} AA(a_j, i)} \tag{5}$$

where $AA(a_j, i)$ is defined as shown in (6):

$$AA(a_j, i) = \begin{cases} 0 \quad \mathit{if}\ MaxADS_{ji} = 0 \\ 1 \quad \mathit{if} \\ \left(\left(MaxADS_{ji} \neq 0\right) \wedge \begin{pmatrix} (s = 2) \vee \\ (\exists l : MaxADS_{ji} = ADS\_List_{ji}[l]) \end{pmatrix} \right) \\ 1 + \left[(s-1) \times MaxADS_{ji} - \sum\limits_{l=1, l \neq j}^{s-1} ADS\_List_{ji}[l] \right] \mathit{otherwise} \end{cases} \tag{6}$$

The *AA* for each of the attributes is calculated using the above formula and the attribute with the highest *AA* score is selected as the fit attribute. According to the above formula, $AA(a_j, i)$, equals zero when the class decision for the rule subset examined corresponds to one class, in that case *MaxADS*$= 0$, which indicates that a leaf node is reached (best case for a branch). $AA(a_j, i)$ equals 1 when s equals 2 or when one of the attributes in the *ADS_list* has an *ADS* score equal to *MaxADS* value (second best case). The second best case indicates that only one extra node will be required to reach a leaf node. Otherwise $AA(a_j, i)$ will be equal to 1 + (the difference between the *ADS* scores of the attributes in the *ADS_list* and the *MaxADS* value) which indicates that more than one node will be required until reaching a leaf node.

4.1.3 Minimum Value Distribution (MVD)

The *MVD* criterion is concerned with the number of values that an attribute has in the current rules. When the highest *AA* score is obtained by more than one attribute, this criterion selects the attribute with the minimum number of values in the current rules. *MVD* criterion minimizes the size of the tree because the fewer the number of values of the attributes the fewer the number of branches involved and consequently the smaller the tree will become [1]. For the sake of simplicity, let us assume that the set of attributes that achieved the highest *AA* score are $a_1, \ldots, a_q, 2 \leq \mathrm{q} \leq \mathrm{s}$. Given an attribute a_j (where $1 \leq j \leq q$), we compute corresponding *MVD* value as shown in (7).

$$MVD(A_j) = \left| \bigcup_{1 \le i \le m} V_{ij} \right| \tag{7}$$

(where |X| denote the cardinality of set *X*).

When the lowest *MVD* score is obtained by more than one attribute, any of these attributes can be selected randomly as the fit attribute. In our experiments in case where more than two attributes have the lowest *MVD* score we take the first attribute.

4.2 Building the Decision Tree

In the decision tree building process, we select the fit attribute that will be assigned to each node from the current set of rules *CR* based on the attribute selection criteria outlined in the previous section. *CR* is a subset of the decision rules that satisfy the combination of attribute values assigned to the path from the root to the current node. *CR* will correspond to the whole set of rules at the root node.

From each node a number of branches are pulled out according to the total number of values available for the corresponding attribute in *CR*.

Each branch is associated with a reduced set of rules *RR* which is a subset of *CR* that satisfies the value of the corresponding attribute. If *RR* is empty, then a single node will be returned with the value of the most frequent class found in the whole set of rules. Otherwise, if all the rules in *RR* assigned to the branch belong to the same decision class, a leaf node will be created and assigned a value of that decision class. The process continues until each branch from the root node is terminated with a leaf node and no more further branching is required.

5 Illustration of the RBDT-1 Method

In this section the *RBDT-1* method is illustrated by a dataset named the weekend problem which is a publicly available dataset.

5.1 Illustration

The Weekend problem is a dataset that consists of ten data records obtained from [12]. For this example, we used the *AQ19* rule induction program to induce the rule set shown in Table 1 which will serve as the input to our proposed method *RBDT-1*. *AQ19* was used with the mode of generating disjoint rules and with producing a complete set of rules without truncating any of the rules. The corresponding

Table 1 The weekend rule set induced by AQ19

Rule#	Description
1:	*Cinema ← Parents-visiting=“yes” & weather = “don’t care” & Money =“rich”*
2:	*Tennis ← Parents-visiting=” “no” & weather =“sunny” & Money =“don’t care”*
3:	*Shopping ← Parents-visiting=“no” & weather =“windy” & Money =“rich”*
4:	*Cinema ← Parents-visiting=“no” & weather =“windy” & Money =“poor”*
5:	*Stay-in ← Parents-visiting=“no” & weather =“rainy” & Money =“poor”*

```
parents-visiting
   yes
      ->    cinema
   no
   weather
      windy
      money
         poor
            ->  cinema
         rich
            ->  shopping
      sunny
         ->    tennis
      rainy
         ->    stay-in
```

Fig. 1 The decision tree generated by RBDT-1 for the weekend problem

decision tree created by the proposed *RBDT-1* method for the weekend problem is shown in Fig. 1. It consists of three nodes and five leaves with 100% classification accuracy for the data.

Figure 2 depicts the tree created by *AQDT-1 & 2* and the *ID3* methods consisting of five nodes and seven leaves, which is bigger than the decision tree created by the *RBDT-1* method. Both trees have the same classification accuracy as *RBDT-1*. Due to space limitations, we are unable to include here all of our work. We refer the reader to our extended technical report on *RBDT-1* [13] for more details on the decision tree generation and discussion on tree sizes.

6 Experiments

In this section, we present an evaluation of our proposed method by comparing it with the *AQDT-1 & 2* and the *ID3* methods based on 16 public datasets. Our evaluation consisted of comparing the decision trees produced by the methods for each dataset in terms of tree complexity (number of nodes and leaves) and accuracy. Other than the weekend dataset, all the datasets were obtained from the UCI machine learning repository [14].

We conducted two experiments for comparing *RBDT-1* with *AQDT-1 & 2* and the *ID3*; the results of these experiments are summarized in Tables 2 and 3. The

Fig. 2 The decision tree generated by AQDT-1, AQDT-2 & ID3 for the weekend problem

```
weather
    windy
    money
        poor
            ->  cinema
        rich
        parents-visiting
            yes
               ->  cinema
            no
               ->  shopping
    sunny
    parents-visiting
        yes
            ->  cinema
        no
            ->  tennis
    rainy
    money
        poor
            ->  stay-in
        rich
            ->  cinema
```

Table 2 Comparision of tree complexities of the RBDT-1, AQDT-1, AQDT-2 & ID3 methods using ID3-based rules

Dataset	Method	Dataset	Method
Weekend	*RBDT-1*	**MONK's 1**	*RBDT-1*
Lenses	*RBDT-1, ID3*	**MONK's 2**	*RBDT-1, AQDT-1 & 2*
Chess	=	**MONK's 3**	=
Car	*RBDT-1, ID3*	**Zoo**	=
Shuttle-L-C	=	**Nursery**	*RBDT-1*
Connect-4	*RBDT-1*	**Balance**	*RBDT-1, ID3*

Table 3 Comparison of tree complexities of the RBDT-1, AQDT-1, AQDT-2 & ID3 methods using AQ-based rules

Dataset	Method	Dataset	Method
Weekend	*RBDT-1*	**Monk's2**	*RBDT-1*
Lenses	*RBDT-1, ID3*	**Monk's3**	=
Zoo	*RBDT-1*	**Chess**	=
Car	*RBDT-1*	**Balance**	*RBDT-1, ID3*
Monk's1	*RBDT-1*	**Shuttle-L-C**	*RBDT-1, AQDT-1 & 2*

results in Table 2 are based on comparing the methods using *ID3-based* rules as input for the rule-based methods. The *ID3-based* rules were extracted from the decision tree generated by the *ID3* method from the whole set of examples of each dataset used. Thus, they cover the whole set of examples (100% coverage). The size of the extracted *ID3-based* rules is equal to the number of leaves in the *ID3* tree.

In Table 3, the experiment was conducted using *AQ-based* rules. We used *AQ-based* rules generated by the *AQ19* rule induction program with 100% correct recognition on the datasets used as input for the rule-based methods under comparison. No pruning is applied by any method in both comparisons presented in Tables 2 and 3.

In the following tables, the name of the method that produced a less complex tree appears in the table in the *method* column; the "=" symbol indicates that the same tree was obtained by all methods under comparison. All four methods run under the assumption that they will produce a complete and consistent decision tree yielding 100% correct recognition on the training examples. For 7 of the datasets listed in Table 2, *RBDT-1* produced a smaller tree than that produced by *AQDT-1 & 2* with an average of 274 nodes less while producing the same tree for the rest of the datasets. On the other hand, *RBDT-1* produced a smaller tree with 5 of these datasets compared to *ID3* with an average of 142.6 nodes less while producing the same tree for the rest of the datasets. The decision tree classification accuracies of all four methods were equal.

Based on the results of the comparison in Table 3, the *RBDT-1* method performed better than the *AQDT-1 & 2* and the *ID3* methods in most cases in terms of tree complexity by an average of 33.1 nodes less than *AQDT-1 & 2* and by an average of 88.5 nodes less than the *ID3* method. The decision tree classification accuracies of all four methods were equal.

7 Conclusions

The *RBDT-1* method proposed in this work allows generating a decision tree from a set of rules rather than from the whole set of examples. Following this methodology, knowledge can be stored in a declarative rule form and transformed into a decision structure when it is needed for decision making. Generating a decision structure from decision rules can potentially be performed much faster than by generating it from training examples.

In our experiments, our proposed method performed better than the other three methods under comparison in most cases in terms of tree complexity and achieves at least the same level of accuracy.

References

1. I. F. Imam, R.S. Michalski, Should decision trees be learned from examples of from decision rules? Source Lecture Notes in Computer Science, in *Proceedings of the 7th International Symposium on Methodologies*, vol. 689. (Springer, Berlin/Heidelberg, 1993), pp. 395–404
2. J.R. Quinlan, Discovering rules by induction from large collections of examples, in *Expert Systems in the Microelectronic Age*, ed. by D. Michie (Edinburgh University Press, Scotland, 1979), pp. 168–201

3. I. H. Witten, B.A. MacDonald, Using concept learning for knowledge acquisition, Int. J. Man Mach. Stud. 27(4), 349–370 (1988)
4. R.S. Michalski, I.F. Imam, Learning problem-oriented decision structures from decision rules: the AQDT-2 system, Lecture Notes in Artificial Intelligence, in *Proceedings of 8th International Symposium Methodologies for Intelligent Systems*, vol. 869 (Springer Verlag, Heidelberg, 1994), pp. 416–426
5. Y. Akiba, S. Kaneda, H. Almuallim, Turning majority voting classifiers into a single decision tree, in *Proceedings of the 10th IEEE International Conference on Tools with Artificial Intelligence*, 1998, pp. 224–230
6. Y. Chen, L.T. Hung, Using decision trees to summarize associative classification rules. Expert Syst. Appl. Pergamon Press, Inc. Publisher, 2009, 36(2), 2338–2351
7. R.S. Michalski, K. Kaufman, The aq19 system for machine learning and pattern discovery: a general description and user's guide, Reports of the Machine Learning and Inference Laboratory, MLI 01-2, George Mason University, Fairfax, VA, 2001
8. J. Wojtusiak, AQ21 user's guide. Reports of the Machine Learning and Inference Laboratory, MLI 04-5, George Mason University, 2004
9. R.S. Michalski, I.F. Imam, On learning decision structures. fundamenta informaticae **31**(1),: 49–64 (1997)
10. R.S. Michalski, I. Mozetic, J. Hong, N. Lavrac, The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, in *Proceedings of AAAI-86*, Philadelphia, PA, 1986, pp. 1041–1045
11. F. Bergadano, S. Matwin, R.S. Michalski, J. Zhang, Learning two-tiered descriptions of flexible concepts: the POSEIDON system. Mach. Learning **8**(1), pp. 5–43 (1992)
12. A. Abdelhalim, I. Traore, The RBDT-1 method for rule-based decision tree generation, Technical report #ECE-09-1, ECE Department, University of Victoria, PO Box 3055, STN CSC, Victoria, BC, Canada, July 2009
13. A. Asuncion, D.J. Newman, UCI machine learning repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, School of Information and Computer Science, Irvine, CA, 2007