**Proceedings of the 2007 IEEE Symposium on**
**Computational Intelligence and Data Mining (CIDM 2007)**

# A Classifier Capable of Handling New Attributes

Dong-Hun Seo, Chi-Hwa Song, Won Don Lee

*Abstract*— **During knowledge acquisition, a new attribute can be added at any time. In such a case, rule generated by the training data with the former attribute set can not be used. Moreover, the rule can not be combined with the new data set with the newly added attribute(s) using the existing algorithms. In this paper, we propose further development of the new inference engine, UChoo, that can handle the above case naturally. Rule generated from the former data set can be combined with the new data set to form the refined rule. This paper shows how this can be done consistently by the extended data expression, and also shows the experimental result to claim the effectiveness of the algorithm.**

## I. Introduction

Ubiquitous computing was introduced as the leader of the third revolution of information by Mark Weiser, in 1989. Since then, the concept of 'ubiquitous' has been recognized as the core of the paradigm of the information industry to come. The governments, the global companies, and the major research labs in the States, Japan, and Europe are developing the technologies related to the 'ubiquitousness' intensively in order to preoccupy the knowledge of this new area and to strengthen the ability in competition.

Ubiquitous environment means the environment where people can get free access to network any time, anywhere, irrespective of time and place, and no matter what devices they use. Mark Weiser indicated three problems for building the ubiquitous environment: First, how to design the chips for low power, Second how to provide the wireless communication network of high bandwidth for the communications among several hundreds of devices. Last, how to interact between user and computer.

Ubiquitous computing environment is meant to be the wireless communication network of many computers which are hidden in the environment. The goal of the Ubiquitous environment is to provide the users with the convenient services. In order to do that, we need to be able to extract the specific information for users out of the daily life as quickly and accurately as possible. Also, we have to be able to give the appropriate information to the users even by inferring the information which was obtained from many various sensors in the ubiquitous environment. This is the identical classification problem that has been dealt in the fields of machine learning and data mining.

In this paper, we use the extended data expression of UChoo, the inference engine proposed by Lee et al. in [1], which Lee suggested based on the classification algorithm of Quinlan, C4.5[2][4][7]. In that paper Lee proposed a method of inference by combining the newly added data into the existing classifier when a new sensor is added to the ubiquitous environment. C4.5 is the classification algorithm which was extended from ID3 and minimizes the usage of memory and shows a fast and good result. In using UChoo, we transform the training data into an extended expression form and construct a decision tree based upon it and generate a rule from the constructed decision tree. We propose an algorithm of inference engine that can systematically construct a decision tree from the data set formed with the newly added sensor.

## II. Extended Training Data Set Expression and Generation of Rule

### A. *UChoo : A New Classifier Based on Extended Data Expression*

Table I shows a general expression of the training data set in C4.5. Table I consists of two discrete attributes, one continuous attribute and a class. The Outlook attribute has three outcomes of sunny, overcast and rain. Windy? has two outcomes of true and false. Both Outlook and Wind? are discrete attributes. And Temp(OF) is a continuous attribute whose values ranges from 60 to 80. Finally, Class has two values with Play and Don't Play.

Table II shows the modified expression of the training data set of Table I. Each entry is to be filled with a probabilistic value range from 0 to 1.

For example, consider Table III. Data in Table III can be made by an expert in this field or can be generated from a decision tree. Below we describe how to fill in the table entry as in Table III using a decision tree in the following section in detail. In case an expert says that it is sunny and the temperature is around 70 and the probability to play is 2/3 no matter it is windy or not, we can express the case as the event #1 shown on the first row of the Table III.

Here we redefine a tuple or instance as the one represented as on row entry with weight value of 1 at the above Table III. In other words, each attribute can have distribution over the possible class values in a tuple or instance. And each row of

Dong-Hun Seo is a Ph.D. candidate at the Department of Computer Science and Engineering at ChungNam National University (e-mail: sm1835dh@hanmail.net)

Chi-Hwa Song is a Professor at the Department of Computer Science and Engineering at ChungNam National University (e-mail: chsong@cnu.ac.kr)

Won Don Lee is a Professor at the Computer Science and Engineering Department, ChungNam National University, DaeJeon, KOREA since 1987. (corresponding author to provide phone: +82 42 821-5448 e-mail: wlee@cnu.ac.kr )

This research is in part supported by Korean post BK program.

TABLE II
Extended data expression tranformed from Table I

| Event# | Outlook | | | Temp($^O$F) | | | Windy? | | Class | |
|---|---|---|---|---|---|---|---|---|---|---|
| | sunny | overcast | rain | 60 | 70 | 80 | True | False | Play | Don't Play |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 6 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

TABLE III
Extended data expression with weights

| Event# | Weight(i) | Outlook | | | Temp($^O$F) | | | Windy? | | Class | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sunny | overcast | rain | 60 | 70 | 80 | True | False | Play | Don't Play |
| 1 | 30 | 1 | 0 | 0 | 0 | 1 | 0 | 1/2 | 1/2 | 2/3 | 1/3 |
| 2 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 6 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

TABLE I
Training Data Set Used in Example

| Outlook | Temp($^O$F) | Windy? | Class |
|---|---|---|---|
| Sunny | 70 | false | Don't Play |
| Sunny | 60 | true | Play |
| Overcast | 80 | false | Play |
| Rain | 60 | true | Don't Play |
| Rain | 70 | false | Play |
| Rain | 80 | true | Don't Play |

Table III has a weight value, which show how much importance the event has. For example, if expert assume that each observed instance has weight of 1 for the importance, the weight value shows how much importance the event has compared with an instance with a weight of 1. The event at the first row of the above Table III has the weight value of 30. In other words, the event has the importance equivalent to that of 30 instances.

Here, an event is a collection of instances with equal attribute value distribution and class value distribution. The event whose weight is 1 can be considered as the instance itself. Therefore, the number of events may not be equal to that of all the instances. For example, in Table III the number of all the instances is 35 while the number of the events is 6. The number of all the instances in a data set T is denoted as |T| and the number of the event is p.

Therefore, we can have newly modified entropy equations by using these weight values as follows.

A is an attribute and k is the number of values of a class and n is the number of outcomes of an attribute. For a continuous attribute, n is 2.

Class membership weight :

$C_1(m), C_2(m), \dots\dots\dots C_{k-1}(m), C_k(m)$.

$C_i(m)$ expresses how much the m th event belongs to the class $C_i$.

Here, $\sum_{i=1}^{k} C_i(m) = 1$

Outcome membership weight :

$O_{A1}(m), O_{A2}(m), \dots\dots\dots O_{A(n-1)}(m), O_{An}(m)$

$O_{Aj}(m)$ is the value which shows how much the outcome value j in the attribute A can happen in the m th event.

Here, $\sum_{j=1}^{n} O_{Aj}(m) = 1$

$T_{Aj}$ : It is the subset of T that has the outcome value j about the attribute A. For example, as the attribute Outlook has three values of sunny, overcast and rain, the set T is divided into three subsets.

p(T) : The number of events in the set T.

Weight(m,T) : Weight value of the m th event in the set T.

freq($C_i$, T) : It is the number of instances in the set T, which have the class value of $C_i$.

In this case,

$$\text{freq}(C_i, T) = \sum_{m=1}^{p(T)} \boldsymbol{Weight(m,T)} \cdot \boldsymbol{C_i(m)}$$

freq($C_i$, $T_{Aj}$) : the number of instances in the set $T_{Aj}$, which have the class value of $C_i$.

$$\text{freq}(C_i, T_{Aj}) = \sum_{m=1}^{p(T)} \boldsymbol{Weight(m,T)} \cdot \boldsymbol{C_i(m)} \cdot \boldsymbol{O_{Aj}(m)}$$

|T| : The number of instances in the set T. An instance means the event whose weight is 1 in here. Therefore, if an event has the weight, W, it means that there are W number of instances with the equal distribution values of attributes and class.

|$T_{Aj}$| : The number of instances in the set $T_{Aj}$.

$$|T_{Aj}| = \sum_{m=1}^{p(T_{Aj})} \boldsymbol{Weight(m,T_{Aj})} \cdot \boldsymbol{O_{Aj}(m)}$$

In this case, by multiplying $O_{Aj}(m)$, the outcome possibility value of each event in $T_{Aj}$, by weight(m,$T_{Aj}$) for each m and adding the results of all the multiplications, we get |$T_{Aj}$|.

TABLE IV
The Extended data expression made from Fig. 3

| Event# | Weight(i) | Outlook | | | Temp($^O$F) | | | Windy? | | Class | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sunny | overcast | rain | 60 | 70 | 80 | True | False | Play | Don't Play |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1/2 | 1/2 | 1 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 | 1/2 | 1/2 | 1/2 | 1/2 | 0 | 1 |
| 3 | 1 | 0 | 1 | 0 | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1 | 0 |
| 4 | 3 | 0 | 0 | 1 | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/3 | 2/3 |

TABLE V
A new training data set

| Evnet# | Weight(i) | Outlook | | | Temp($^O$F) | | | Windy? | | Class | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sunny | overcast | rain | 60 | 70 | 80 | True | False | Play | Don't Play |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 3 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

Therefore, we can decide the attribute which has the biggest Gain_ratio in the node by using the existing entropy equations with these newly defined values. Here,

Gain_ratio(A) = Gain(A) / Split_info(A) is used as in C4.5,and the Gain(A) is the mutual information between the class and the attribute A, and the Split_info(A) is the normalization constant so that the number of the outcomes of the attribute A do not affect in the information measure. Refer [1] how UChoo builds the decision tree using above definitions and the information measures.

## *B. Generation of the Rule*

The rule refinement problem is to construct a new tree by adding the new data into the rule which is constructed from the original data set.

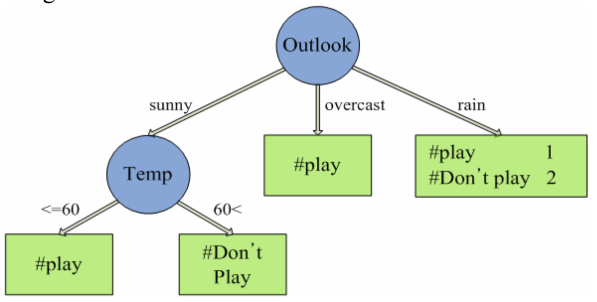

Fig. 1. Decision tree

Let the original data set be as in Table I. Table II shows the same data set in the extended data expression. Using the set, UChoo generates the tree as in Fig. 3 using the equations described in the section 2.1. From Fig. 3, the rule can be extracted as in Table IV. Now, assume that a new data set is collected as in Table V. UChoo can construct a new decision tree. This is consistent with the generation of the decision tree from the training data only, as there is no difference in the data representation between the training data and the rule. Note, for instance, that in the extended data expression a leaf node of the decision tree with m C1 events and n C2 events can be described as if a data event with the weight of (m+n) and with probability of being C1 as m/(m+n) and with probability of being C2 as n/(m+n). In other words, the suggested extended data expression deals with the rule and the individual event coherently and therefore makes no difference in using and treating them in making the decision tree. This feature is so nice and advantageous that we do not have to worry about how to refine the decision tree with the heuristics. This kind of approach can not only be used in the decision tree-like classifier, but also can be used as in other forms of the classifier as well. This is, as far as we are aware of, is the first attempt of its kind in the rule refinement research area. We not only do not have to use the heuristics to discriminate between the expert driven rule and the training data, but also can use the inference engine algorithm consistently in rule refinement. The communication between the expert and the machine can be natural as the algorithm does not discriminate between the two rules from the machine and the expert. The thing left is how to control or assign the importance of the expert driven rule compared with the training data or the machine generated rule. This is again the traditional design problem that an engineer encounters. There is, however, a measure of how important the machine generated rule is, i.e., the number of the data events belonging to the leaf nodes of the decision tree can be a natural estimate of the importance of the rule when one considers each event having the default importance value of one.

## III. Experiment

Assume that we have one original data set with n attributes. And as times goes by, a new sensor is added to the network, and therefore new attributes are added to the already existing attributes. The data collected with these attributes constitutes a new data set to refine the former rule generated from the original data set.

We perform the experiment using the data sets collected at UCI Machine Repository[6]. Those data sets do not include missing or unknown value. We assume that we know the range of the newly added attributes' values.

The data in our experiment is divided into 10 blocks. Among them, 9 blocks are for the training and the rest one is for testing. And the testing data set is used to measure the

TABLE VIII

Extended data expression with a new attribute

| Event # | Weight(i) | Outlook | | | Temp($^0$F) | | | Windy? | | Homework | | Class | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sunny | overcast | rain | 60 | 70 | 80 | true | false | O | X | Play | Don't Play |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1/2 | 1/2 | 1/2 | 1/2 | 2/3 | 1/3 |
| 2 | 1 | 1 | 0 | 0 | 0 | 1/2 | 1/2 | 1 | 0 | 1/2 | 1/2 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 | 1/3 | 1/3 | 1/3 | 0 | 1 | 1/2 | 1/2 | 1 | 0 |
| 4 | 3 | 0 | 0 | 1 | 1/3 | 1/3 | 1/3 | 1 | 0 | 1/2 | 1/2 | 1/3 | 2/3 |
| 5 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 6 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 7 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

error rate. We try to make each block having the similar distribution of the classes with that of the classes in the total data set. This is called as the 10-fold cross validation. Each of the UCI data set is divided into 10 blocks: 9 blocks for the training, 1 block for the testing. After making the 9 blocks, those 9 blocks are again divided randomly into 2 blocks, Training1 and Training2 as in Fig. 1. These two blocks are finally used as the training data set for UChoo.

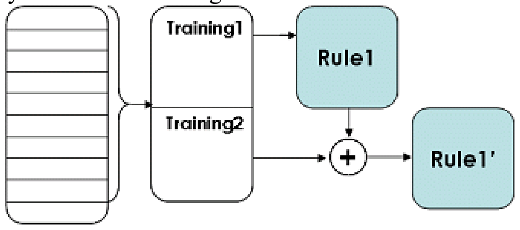

Fig. 2. Rule refinement

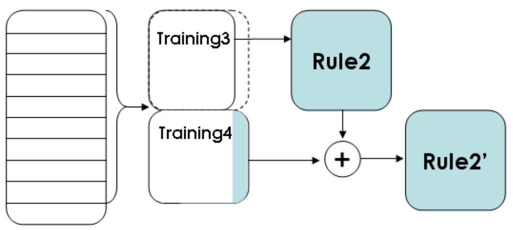

Fig. 3. Rule refinement when new attributes are added

For the experiment of the rule refinement, we first generated a rule, Rule1 as in Fig. 1, using the Training1. By adding the Training2 to the Rule1, a new rule, Rule1' in Fig. 1, is generated. Note that the Rule1 and the Training2 are described in the same way in the extended data expression. The result is shown in the column UChoo in Tables 6 and 7. Note that in the Letter data C4.5 is better, but in the Iris data UChoo outperforms C4.5. Here, note that C4.5 uses all the raw data, Training1 and Training2, to generate the rule, whereas UChoo used the rule generated only the Training1 and the data Training2 to generate the refined rule, Rule1', as in Fig. 1. If UChoo generated the rule from the raw data, Training1 and Training2, then the result would have been the same as C4.5. Now, however, as UChoo generated refined rule from the Rule1 and Training2, the performance should not be better than the case when generating the rule directly from the raw data, as the information content of the Rule1 might be less than or equal to that of the raw data. As we know, the rule is a generalization of the example data set and therefore loses some of the possible information contained in the original data set and therefore the information content of the rule should be always less than or equal to that of the raw data set. In that sense, the fact that the performance of the UChoo is comparable to or outperformed C4.5 indicates that UChoo truly is a powerful inference engine in doing the classification. One might wonder why the performance of the UChoo is sometimes even better than that of the C4.5, if the information content of the Rule1 is less than that of the Training1. The thing is that Rule1 can be used as the 'guidance' to lead to the main hyperspace for a given class and the Training2 can be used in making the chosen space in detail. In other words, the so called problem of 'overlearning' can be avoided as we refine if the former rule, instead of the former data set, is used in making a new rule. In that way, a rule generated by UChoo can be more efficient than that generated by C4.5.

Now, to experiment the case with the new attribute(s), we remove an attribute from Training1 and name it as Training3. From Training3, Rule2 is made by UChoo as in Fig. 2. Table VIII shows an example case when a new attribute 'Homework' is added to the former attributes. The old data set consists 4 events, numbered from 1 to 4, and there are only three attributes, Outlook, Temp and Windy?. As the old data set does not have the new attribute 'Homework' its value is treated as 'don't care' values, and hence the Homework outcomes have values of 1/2 as there are two outcomes. The new data events are numbered from 5 to 7, and note that the data set, either old or new, are all in the same form, naturally represented in the extended data expression form. Using this table, UChoo can generate the refined rule systematically.

The column Training3 in Tables 6 and 7 indicate the error rate of this case. Note that the error rate if pretty high as one attribute is missing. It shows that the attribute has some mutual information with the class and therefore is not a useless one. Traing4, having all the attributes, therefore has one more attribute than the Training3. The Rule2 is combined with the Training4 to form the Rule2' by UChoo. The performance of the Rule2' thus generated is shown in the column 'Training4 + Rule2' in Table VI and 7. Note that the performance of the 'Training4 + Rule2' is better than 'UChoo' in Letter data case and worse in Iris case, though the differences are minute. In this case, when a rule made from the already existing attributes is combined with the newly added training set, the refined rule is still efficient in making decisions. In this way, we do not have to throw away the former 'experience', and still can use those former information. As ubiquitous computing environment gets more attention, many new sensors will be added to the network. Each time new sensors are added, we do not have throw away the old data set, simply because it lacks some sensors, but still can utilize it. Also, when the size of the collected new data set is small, the rule generated from the new data set only will not have high performance. Therefore, utilizing the former data set or the former rule becomes important. Also, when the former data set is lost or should be thrown away because of the memory limitation, using the former rule becomes important. As we mentioned already above, the 'guidance' role of the rule becomes sometimes important and the feature UChoo has is actually becomes powerful in that sense.

TABLE VI

Letter data

| # Total Set | # Training | Testing | | Error( % ) | | | C4.5 |
|---|---|---|---|---|---|---|---|
| | | | | UChoo | Training3 | Training4 +Rule2 | Error(%) |
| | | | 0 | 27.30 | 30.60 | 27.50 | 27.55 |
| | | | 1 | 26.65 | 28.80 | 27.10 | 24.90 |
| | | | 2 | 26.55 | 30.40 | 27.50 | 27.80 |
| | | | 3 | 26.69 | 31.65 | 26.85 | 24.70 |
| | | | 4 | 29.90 | 30.15 | 29.45 | 26.15 |
| | | | 5 | 28.95 | 31.55 | 27.60 | 25.60 |
| | | | 6 | 30.55 | 31.60 | 28.60 | 24.45 |
| | | | 7 | 26.35 | 31.75 | 26.80 | 25.50 |
| | | | 8 | 28.75 | 30.35 | 27.00 | 28.60 |
| | | | 9 | 31.35 | 31.70 | 29.10 | 25.90 |
| | | | | 28.304 | 30.885 | 27.75 | 26.115 |

TABLE VII

Iris data

| # Total Set | # Training | Testing | | Error( % ) | | | C4.5 |
|---|---|---|---|---|---|---|---|
| | | | | UChoo | Training3 | Training4 +Rule2 | Error(%) |
| | | | 0 | 6.67 | 6.67 | 6.67 | 6.77 |
| | | | 1 | 6.67 | 20.00 | 0.00 | 0.00 |
| | | | 2 | 13.33 | 20.00 | 6.67 | 13.33 |
| | | | 3 | 26.67 | 26.67 | 20.00 | 33.33 |
| | | | 4 | 0.00 | 40.00 | 20.00 | 0.00 |
| | | | 5 | 0.00 | 20.00 | 0.00 | 0.00 |
| | | | 6 | 13.33 | 33.33 | 6.67 | 13.33 |
| | | | 7 | 6.67 | 20.00 | 6.67 | 20.00 |
| | | | 8 | 6.67 | 20.00 | 6.67 | 6.67 |
| | | | 9 | 0.00 | 20.00 | 13.33 | 6.67 |
| | | | | 8.001 | 22.667 | 8.668 | 10.01 |

## IV. CONCLUSION

In this paper, a method of a new inference engine is proposed that is capable of not only making a decision tree for a given data set, but also making one with the already generate rule and the data set. One can easily see that this algorithm also can generate the rule from many old rules only, without any data set. It becomes possible because the rule and the data set are represented in the same format, as suggested in the extended data expression form. Experiment shows that this method is not only natural in making the expert and the machine communicate coherently, but also powerful enough to be used for a classifier.

Another important issue dealt with is that when new sensors are added to the modern ubiquitous computing environment, and hence new attributes are added or deleted, the algorithm naturally generates the rule using the information available. The rule thus made is shown to be comparable with that generated from the raw data set. Although memory limitation is set or when the data is lost, the algorithm still can utilize the information contained in the formerly generated rule systematically.

In the future, we would like to test the case when newly added attribute has a non-uniform distribution. Also, finding out the right way to prune the decision tree will be another issue to examine closely.

## REFERENCES

[1] Dong-Hui Kim, Dong-Hyeok Lee and Won Don Lee, "Classifier using Extended Data Expression", IEEE Mountain Workshop on Adaptive and Learning Systems, pp. 154-159, July. 2006
[2] J. R. Quinlan, "C4.5:Program for Machine Learning" ,San Mateo, Calif, Morgan Kaufmann, 1993.
[3] T. S. Lim, W. Y. Loh, and Y. S. Shih, "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Tree Old and New Classification Algorithms", Machine Learning, vol. 40, no. 3, pp. 203-228, Sept.2000.
[4] J. R. Quinlan, "Bagging, Boosting, and C4.5," AAAI/IAAI , vol. 1, 1996.
[5] Pang-Ning Tan, Michael SteinBach, Vipin Kumar,"Introduction to DATA MINING", Addison Wesely, pp. 207-312, 2005
[6] http://www.ics.uci.edu/~mlearn/MLRepository.html, UCI Machine Learning, 1998
[7] Ronny Kohavi, J. R. Quinlan, "Data mining tasks and methods: Classification: Decision-tree discovery", Handbook of data mining and knowledge discovery, Oxford University Press, pp. 267-276, 2002.