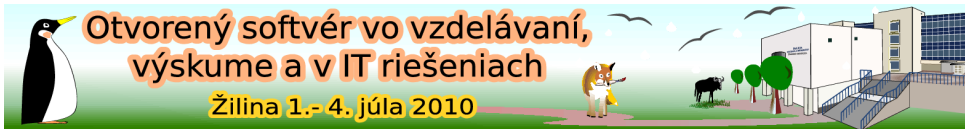

WEKA AS A TOOL FOR CLASSIFICATION TASKS

BOHÁČIK, Ján, (SK)

Abstract. Classification belongs to the main tasks in the Data Mining stage of the Knowledge Discovery in Databases. The aim of this paper is to compare several classification data mining algorithms using the Weka software tool implemented in Java and issued under the GNU General Public License. Some of results of the author are also described.

1 Introduction

Data repositories are expanding very quickly and contain immense amounts of various data. According to some estimates these amounts doubling every twenty months [4]. Due to it, information and database systems are widely used. The current generation of database systems is based mainly on a small number of primitives of Structured Query Language (SQL) used for manipulating with relational databases which consist of relations (tables) [7]. The tables have attributes in columns and instances in rows. Often, these tables contain more and more covert information that cannot be found out and transformed into knowledge by classical SQL queries. Dependencies are searched so that knowledge can be extracted. There are several ways how to use found dependencies, which classification is one of the most used ones from.

Classification is often solved in the Data Mining stage of the Knowledge Discovery in Databases. Given a set of training instances (known instances) $\boldsymbol{V}$ where each instance $\boldsymbol{e} \in \boldsymbol{V}$ is described by attributes $\boldsymbol{A} = \{A_1, \ldots, A_k, \ldots, A_M\}$, $A_k = \{a_{k,1}, \ldots, a_{k,l}, \ldots, a_{k,N_k}\}$, $a_{k,l}$ is a possible value of the attribute, and classified into a class $c_j \in C$ where $C = \{c_1, \ldots, c_j, \ldots, c_O\}$ is a set of possible classes, the task is to build a model that predicts the class of an unseen instance. This model can be used

in systems where a decision is required on the basis of known data. Systems of this kind are described more in [11, 12]. An example of using in transportation is a prediction of accidents on the basis of risk factors [3]. Among most popular classification models are the Nearest Neighbor Classifier model, the Naive Bayes Classifier model and the Fuzzy Decision Tree model. These models are compared in this paper on the basis of their error rates in classification. A modification of using the Fuzzy Decision Tree model is described. The algorithms for making the first two models are implemented in Weka [13] – an object-oriented machine-learning Java software tool issued under the GNU General Public License. The algorithm for making the Fuzzy Decision Tree model is implemented in my object-oriented Java software tool Fuzzy Rule Miner [2].

The paper is organized as follows. Section 2 explains object-oriented Java tool Weka. In Section 3, principles of building the Nearest Neighbor Classifier model and the Naive Bayes Classifier model are stated. The principle of building the Fuzzy Decision Tree model and two ways how to use it are described in Section 4. The results of experiments are in Section 5. Section 6 concludes this paper.

2 Weka

Weka (http://www.cs.waikato.ac.nz/ml/weka/, Waikato Environment for Knowledge Learning) is a machine-learning tool developed at the Waikato University in the New Zealand [13]. Originally written in C, Weka has been completely rewritten in Java and is compatible with almost every computing platform. It can cope with pre-processing and data analysis, classification models, association models, and evaluation metrics. There are three modes of Weka operation: a) GUI, b) command-line and c) Java API. Java API allows to make computer programs for solving classification tasks. In the following lines, there is a code showing how to make attributes and instances:

```
// Declare an attribute with numerical values:
  Attribute A1 = new Attribute("A1");
// Declare an attribute with nominal values:
  FastVector A2NomVal = new FastVector(2);
  A2NomVal.addElement("a2,1");
  A2NomVal.addElement("a2,2");
  Attribute A2 = new Attribute("A2", A2NomVal);
// Declare the class attribute:
  FastVector CVal = new FastVector(2);
```

```
  CVal.addElement("c1");
  CVal.addElement("c2");
  Attribute C = new Attribute("C", Cval);
// Declare the vector of attributes A and class C:
  FastVector AandC = new FastVector(3);
  AandC.addElement(A1);
  AandC.addElement(A2);
  AandC.addElement(C);
// Create training instances V:
  Instances V = new Instances("V", AandC, 10); // 10 is the initial capacity
  V.setClassIndex(2);
// Create an instance and add it:
  Instance e1 = new Instance(3);
  e1.setValue((Attribute)AandC.elementAt(0), 1.0);
  e1.setValue((Attribute)AandC.elementAt(1), "a2,2");
  e1.setValue((Attribute)AandC.elementAt(3), "c1");
  V.add(e1);
```

In the *weka.classifiers* package, the most important class is *Classifier*. It is a general scheme for any classification model in Weka. *Classifier* contains two significant methods, *buildClassifier()* and *classifyInstance()*. The former is for building the classification model, the latter is for determining the value of class attribute C when the values of all attributes in $\boldsymbol{A}$ are known (i.e. classification).

3 Nearest Neighbor Classifier and Naive Bayes Classifier

Nearest Neighbor Classifier (NNC) assumes that all instances correspond to points in the n-dimensional space $\boldsymbol{R}^n$ [8]. NNC is a type of lazy learning where the function is only approximated locally and all computation is deferred until classification. During learning, all points with known classes are remembered. When a new point is classified, the k-nearest points to the new point are found (k is a positive integer, usually small) and are used with a weight for determining the class of the new point (the instance with unknown class). For the sake of increasing classification accuracy, greater weights are given to closer points during classification. The simple version of the algorithm is easy to implement by computing the distances from the instance with unknown class to all stored known instances $\boldsymbol{V}$, but it is computationally intensive, especially when the cardinality of $\boldsymbol{V}$ grows. Many nearest neighbor classifier models have been proposed; these

generally seek to reduce the number of distance evaluations. Its implementation in Weka is in the *weka.classifiers.lazy* package and it can be used as follows:

```
// Create a new Nearest Neighbor Classifier:
Classifier NNC = (Classifier)new IBk();
NNC.buildClassifier(V);
```

Naive Bayes Classifier (NBC) is a probabilistic classification model based on Bayes' theorem. It represents each instance in $\boldsymbol{V}$ as a #($\boldsymbol{A}$)-dimensional vector of attribute values $[a_{1,l_1}, a_{2,l_2}, \ldots, a_{M,l_M}]$, #($\boldsymbol{A}$) is the cardinality of $\boldsymbol{A}$. Given that there are O classes $c_1, \ldots, c_j, \ldots, c_O$, the classifier predicts that an unknown instance with known values of vector $X = [x_1, \ldots, x_k, \ldots, x_M]$) belongs to the class c_j having the highest posterior probability conditioned on X. In other words, X is assigned to class c_j if and only if $P(c_j/X) > P(c_k/X)$ for $1 \leq k \leq O$ and $j \neq k$. NBC is very simple, it requires only a single scan of the data, thereby providing high accuracy and speed for large databases. However, inaccuracies arise due to a) the simplified assumptions involved and b) a lack of available probability data or knowledge about the underlying probability distribution [9, 13]. Its implementation in Weka is in the *weka.classifiers* package and it can be used as:

```
// Create a new Naive Bayes Classifier:
Classifier NBC = (Classifier)new NaiveBayes();
NBC.buildClassifier(V);
```

4 Fuzzy decision tree classification with more leaf nodes

In this section, two ways how to use a fuzzy decision tree (FDT) for determining the class of an instance $\boldsymbol{e}$ are described. The FDT is made with a top-down greedy ID3-like heuristic based on [6]. Before it is built, attributes are transformed into linguistic terms $\boldsymbol{A} = \{A_1, \ldots, A_k, \ldots, A_M\}$, $A_k = \{a_{k,1}, \ldots, a_{k,l}, \ldots, a_{k,N_k}\}$, $a_{k,l}$ is a linguistic term of the linguistic variable A_k, and classes are transformed into class linguistic terms $c_j \in C$, $C = \{c_1, \ldots, c_j, \ldots, c_O\}$ is the class linguistic variable. For each $a_{k,l} \in A_k \in \boldsymbol{A}$, $c_j \in C$ and $\boldsymbol{e} \in V$, membership degree $a_{k,l}(\boldsymbol{e})$ ($c_j(\boldsymbol{e})$) to which $a_{k,l}$ (c_j) is the value of A_k (C) for instance $\boldsymbol{e}$ is determined. The value of a membership degree is in the continuous interval 0 to 1. Number 1 means the highest possibility that $a_{k,l}$ (c_j) is the value of A_k (C), number 0 means the lowest possibility that $a_{k,l}$ (c_j) is the value of A_k (C). The transformations and computing membership degrees can be done for example with algorithm [5]. During building, main activities are the association of a linguistic variable with a node and the decision if a node or a leaf should be associated with a branch coming from a

node. The former is done with computing cumulative information as a criterion. The linguistic variable with the maximal cumulative information is chosen. The latter uses a stopping criterion based on the frequency of the branch. The FDT is made as follows. The root node is made. Linguistic variable $maxA_k$ is associated with it after cumulative information is computed for all linguistic variables in $\boldsymbol{A}$. A branch coming from the root node is made for each $a_{k,l} \in maxA_k$ and it is associated with the $a_{k,l}$. These branches are considered unprocessed. For each unprocessed branch, the decision if an internal node or a leaf node is connected with it is made. Then the branch is considered processed. If an internal node was connected with it, a linguistic variable is associated with it. The process of building finishes when all branches are considered processed. Let the FDT have R leaf nodes $\boldsymbol{L} = \{l_1, \ldots, l_r, \ldots, l_R\}$. During building the FDT, also value F_j^r for each leaf node l_r and each linguistic term c_j is made. This value F_j^r means the certainty degree of the linguistic term c_j attached to the leaf node l_r.

Classification of an instance $\boldsymbol{e}$ using the FDT means determining the values of membership degrees $c_j(\boldsymbol{e})$ for all $c_j \in C$. It is also supposed that membership degrees $a_{k,l}(\boldsymbol{e})$ for all $a_{k,l} \in A_k \in \boldsymbol{A}$ are known. The first way how to conduct classification is based on [10]. It was initially meant for classification in the crisp case. In the crisp space, a linguistic term either is the value of a linguistic variable or it is not. In other words, either $a_{k,l}(\boldsymbol{e})=1$ or $a_{k,l}(\boldsymbol{e})=0$ for all $a_{k,l} \in A_k \in \boldsymbol{A}$. Also, just one $a_{k,l}(\boldsymbol{e})=0$ for all $a_{k,l} \in A_k$, all the others equal 0. In the fuzzy case, there can be more than one $a_{k,l}(\boldsymbol{e})>0$ for all $a_{k,l} \in A_k$. For the purpose of using this first way, $a_{k,l}(\boldsymbol{e})$ with the maximal value among all $a_{k,l} \in A_k$ is set to 1 and all the others are set to 0. The instance $\boldsymbol{e}$ with these membership degrees is called rounded instance $\boldsymbol{e}$. In the FDT, a path from the root node to the leaf node l_r where terms a_{k_1,l_1}, a_{k_2,l_2}, …, a_{k_q,l_q} associated with the branches in the path respectively match with $a_{k,l}$ with $a_{k,l}(\boldsymbol{e})=1$ of the rounded instance $\boldsymbol{e}$. After that, $c_j(\boldsymbol{e})=F_{j_q}^r$.

The other way for classification of an instance $\boldsymbol{e}$ uses one or more paths from the root node to the leaf nodes. The reason why it is like this is because there may be several $a_{k,l}(\boldsymbol{e})>0$ for a node associated with A_k in the FDT. That is why there may be several paths, from the root node to the leaf nodes, whose all branches coming from the nodes are associated with $a_{k,l}$ with $a_{k,l}(\boldsymbol{e})>0$. Because all $a_{k,l}(\boldsymbol{e})$ do not equal 1 in general in the path, it is clear that each path of this kind should be included in the final value of $c_j(\boldsymbol{e})$ with a certain weight. It is defined for instance $\boldsymbol{e}$ and the path from the root node to the leaf node l_r as follows:

$$W_r(\boldsymbol{e}) = \prod_{a_{k,l} \in \boldsymbol{PATH}_r} a_{k,l}(\boldsymbol{e}), \tag{1}$$

where $\boldsymbol{PATH}_r$ is a set of all linguistic terms $a_{k,l}$ associated with the branches in the path from the root node to the leaf node l_r. Membership degrees $c_j(\boldsymbol{e})$ for all $c_j \in C$ are computed as follows:

$$c_j(\boldsymbol{e}) = \sum_{r=1}^{R} F_j^r \cdot W_r(\boldsymbol{e}). \tag{2}$$

If, in the first or the second way, classification only into one class linguistic term $c_j \in C$ is required, instance $\boldsymbol{e}$ is classified into $c_j \in C$ whose $c_j(\boldsymbol{e})$ is maximal. Software tool Fuzzy Rule Miner of the author contains similar classes to Weka.

5 Experiments

The main purpose of the experiments is to compare the two ways how to use a fuzzy decision tree based on [6] (FDT) for determining the class of an instance $\boldsymbol{e}$ with each other and with Nearest Neighbor Classifier (NNC) and Naive Bayes Classifier (NBC). The algorithm for building the FDT and the two ways of determining the class of an instance $\boldsymbol{e}$ are implemented in object-oriented Java tool Fuzzy Rule Miner of the author [2]. The NNC and NBC are implemented in Weka [13] – an object-oriented machine-learning Java software tool issued under the GNU General Public License.

The experiments are carried out on selected machine-learning databases [1]. First, if it was required the databases were fuzzified according to [5]. Then each database was randomly separated into two parts. One part contained 70% of the database and it was used for building classification models. The other part contained 30% of the database and it was used for verification of the classification models. This process of separation and verification was repeated 100 times. Partial error rates for respective databases were obtained. The error rate was finally calculated as the ratio of the number of mis-classification combinations to the total number of combinations.

The results of the experiments are in Table 1 where FDT-M denotes fuzzy decision tree based on [6] and the second way of determining the class of an instance $\boldsymbol{e}$ stated in Section 4, FDT-O denotes fuzzy decision tree based on [6] and the first way of determining the class of an instance $\boldsymbol{e}$ stated in Section 4, NNC denotes Nearest Neighbor Classifier and NBC denotes Naive Bayes Classifier. The last row contains average error rates for all databases and respective classification models. It can be seen that FDT-M which uses several paths from the root node to

leaf nodes for classification is almost 10% better that FDT-O and it is the best of all classification models compared here.

Table 1: Error rates for respective classification models

Database	**Error rate for a database and a classification model**			
	FDT-M	**FDT-O**	**NNC**	**NBC**
BUPA	0.4174	0.4205	0.3910	0.4416
Ecoli	0.2022	0.2654	0.2046	0.1547
Glass	0.3988	0.4647	0.3335	0.5394
Haberman	0.2624	0.2609	0.3471	0.2453
Iris	0.04067	0.02956	0.05044	0.04556
Pima	0.2436	0.2563	0.3091	0.2483
Wine	0.04566	0.06509	0.05094	0.02697
Average	**0.2301**	**0.2518**	**0.2409**	**0.2431**

6 Conclusions

In this paper, object-oriented machine-learning Java software tool Weka was described with focus on classification models and classification tasks. Mechanisms of using a fuzzy decision tree for determining class linguistic terms of instances were investigated. Two ways were described. One is an analogy of the crisp case when only one path from the root node to a leaf node is used. The other is a generalization of the former where several paths are taken into consideration with certain weights. The generalized way has almost 10% better results on selected databases and also it is the best of all compared classification models.

7 References

[1] ASUNCION, A., NEWMAN, D., J.: UCI Machine Learning Repository [`http://www.ics.uci.edu/mlearn/MLRepository.html`], Irvine, University of California, Department of Information and Computer Science, CA, 2007.

[2] BOHACIK, J., MATIASKO, K., LEVASHENKO, V.: Software for making and using fuzzy rules, Journal of ELECTRICAL ENGINEERING, Vol. 57, pp. 85-88, 2006, ISSN 1335-3632
[3] CHANG, L.-Y., CHEN W.-C.: Data mining of tree-based models to analyze freewayaccident frequency, Journal of Safety Research, Vol. 36, No. 4, pp. 365-375, 2005, ISSN 0022-4375.
[4] JOHN, G. H.: Enhancements to the Data Mining Process, Standford: PhD Thesis, 1997, Standford University
[5] LEE, H.-M., CHEN, C.-M., CHEN, J.-M., JOU, Y.-L.: An efficient fuzzy classifier with feature selection based on fuzzy entropy. IEEE Transaction on Systems, Man, and Cybernetics – Part B: Cybernetics 3, Vol. 31, pp. 426-432, 2001
[6] LEVASHENKO, V., ZAITSEVA, E.: Usage of new information estimations for induction of fuzzy decision trees. Proc of the 3rd IEEE International Conference on Intelligent Data Engineering and Automated Learning, Kluwer Publisher, UK, pp. 493-499, 2002
[7] MATIASKO, K.: Databazove systemy (Database systems), Zilina: University of Zilina, 2009, ISBN 80-7100-968-7
[8] MITCHELL, T. M.: Machine Learning. USA: McGraw-Hill Companies, 1997, ISBN 0-07-042807-7
[9] MITRA, S., ACHARYA, T.: Data mining – multimedia, soft computing, and bioinformatics. NJ: Wiley, 2003, ISBN 0-471-46054-0
[10] QUINLAN, J. R.: Induction of decision trees. Machine Learning, No. 1, pp. 81-106, 1986
[11] TRNKA, A.: Proposal of application datawarehouses into control process. In: Proc. of the International Doctoral Seminar, AlumniPress, Smolenice, pp. 343-348, 2009, ISBN 978-80-8096-088-9
[12] TRNKA, A., TANUSKA, P.: Datawarehousing and data mining methodologies as a one of the possible benefits in control process. In: Aktualnye problemy i innovacii v ekonomike, upravlenii, obrazovanii, informacionnych technologijach : materialy mezdunarodnoj naucnoj konferencii (12-15 maja 2009 goda, Stavropol-Kislovodsk) - Vol. 5, No. 3, pp. 84-87., 2009, ISSN 2074-1685.
[13] WITTEN, I. H., FRANK, E.: Data mining: Practical machine learning tools and techniques (2nd edition). San Francisco: Morgan Kaufman, 2005, ISBN 0-12-088407-0

Contact address

Ján BOHÁČIK (Ing., PhD.), Department of Informatics, FRI ZU in Žilina, Univerzitná 8215/1, 010 26 Žilina, Jan.Bohacik@fri.uniza.sk

This work is made with the support of

Center of excellence for systems and services of intelligent transport

ITMS code of the project 26220120028

University of Zilina in Zilina

ERDF - European Regional Development Fund

Project is financially supported by the sources of EC

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
1.–4. júla 2010, Žilina, Slovensko

Organizátori: Miloš Šrámek, Spoločnosť pre otvorené informačné technológie
Tatiana Šrámková, Katedra fyziky, FEI STU Bratislava
Michal Kaukič, Aleš Kozubík, Tomáš Majer, Žilinská univerzita
Lýdia Gábrišová, Ľubica Michálková, Žilinská univerzita
Juraj Bednár, Digmia, Slovensko
Miloslav Ofúkaný, GeoCommunity, Slovensko
Peter Mráz, Kremnica
Slavko Fedorik, SOŠ elektrotechnická, Poprad
Peter Štrba, Spojená škola/Gymnázium M. Galandu, Turčianske Teplice
Ladislav Ševčovič, FEI, Technická univerzita v Košiciach

Editori: Michal Kaukič
Miloš Šrámek
Slavko Fedorik
Ladislav Ševčovič

Recenzenti: Mgr. Juraj Bednár
Mgr. Rudolf Blaško, PhD.
RNDr. Ján Buša, CSc.
Ing. Slavko Fedorik
Ing. Karol Grondžák, PhD.
Mgr. Michal Kaukič, CSc.
Ing. Tomáš Kliment
RNDr. Aleš Kozubík, PhD.
Mgr. Juraj Michálek
doc. RNDr. Štefan Peško, CSc.
Ing. Pavel Stříž, PhD.
RNDr. Ladislav Ševčovič
Ing. Michal Žarnay, PhD.

Vydavateľ: Spoločnosť pre otvorené informačné technológie – SOIT, Bratislava

ISBN 978-80-970457-0-8

Sadzba programom pdfTEX Ladislav Ševčovič
