Integrating an Advanced Classifier in WEKA

Paul Ştefan Popescu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
sppopescu@gmail.com

Mihai Mocanu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
mocanu@software.ucv.ro

Marian Cristian Mihăescu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
mihaescu@software.ucv.ro

ABSTRACT

In these days WEKA has become one of the most important data mining and machine learning tools. Despite the fact that it incorporates many algorithms, on the classification area there are still some unimplemented features. In this paper we cover some of the missing features that may be useful to researchers and developers when working with decision tree classifiers. The rest of the paper presents the design of a package compatible with the WEKA Package Manager, which is now under development. The functionalities provided by the tool include instance loading, successor/predecessor computation and an alternative visualization feature of an enhanced decision tree, using the J48 algorithm. The paper presents how a new data mining/machine learning classification algorithm can be adapted to be used integrated in the workbench of WEKA.

Keywords

Classifier, J48, WEKA, Machine learning, Data Mining

1. INTRODUCTION

Nowadays huge amounts of data can be gathered from many research areas or industry applications. There is a certain need for data mining or knowledge extraction [6] from data. From this large amount of data, the data analysts gather many variables/features and many machine learning techniques are needed to face this situation. There are many application domains such as medical, economics (i.e., marketing, sales, etc.), engineering or in our case educational research area [16] in which machine learning techniques can be applied. Educational data mining is a growing domain [4] in which a lot of work has been done.

Because the application domains are growing continuously, the tools that support the machine learning processes must live up to market standards providing good performances and intuitive visualization techniques. In these days there are many tools that deal with a wide variety of problems. In order to be more explicit, we have tools like RapidMiner [12], KEEL [2], WEKA, Knime [3] or Mahout [13]. RapidMiner is a graphical drag and drop analytics platform, formerly known as YALE, which provides an integrated environment for data mining, machine learning, business and predictive analytics. Keel is an application package of machine learning software tools, specialized on the evaluation of evolutionary algorithms. KNIME, the Konstanz Information Miner, is a modular data exploration platform, provided as an Eclipse plug-in, which offers a graphical workbench and various components for data mining and machine learning. Mahout is a highly scalable machine learning library based on the Hadoop framework [18], an implementation of the MapReduce programming model, which supports distributed processing of large data sets across clusters of computers.

For our approach we choose WEKA because it has become one of the most popular machine learning and data mining workbenches and its success is due to its constant improvement and development. Moreover, WEKA is a very popular tool used in many research domains, widely adopted by the educational data mining communities.

WEKA is developed in Java and encapsulates a collection of algorithms that tackle many data mining or machine learning tasks like preprocessing, regression, clustering, association rules, classification and also visualization techniques. In some cases, these algorithms are referring only the basic implementation.

One aspect that needs to be taken into consideration is that WEKA has a package manager which simplifies the developers contribution process. There are two kind of packages that can be installed in WEKA and used via the application interface: official and unofficial packages. This is a very important feature because if there is an algorithm that fits your problem description and there is a package for it you can just add it to the application and use it further. Moreover, you don't need to be a programmer to do that, you don't need to write code, just install the package and then use the algorithm like it had been there forever.

According to the real life experiences, many of the included algorithms can hardly be used because of their lack of flexibility. For example, in standard decision trees from WEKA we can perform a classification process but we cannot access a particular instance from the tree. Suppose that we have a training data file and we create the tree model. When we try

to see where is the place of the instance "X" in the tree we can't do that in the application interface, neither when you add the WEKA library in your code. This is a big drawback because retrieving the leaf to which the instance belongs to provides more information than retrieving its class. Usually, when performing a classification task, the data analyst divides test instances into classes that have little meaning from application domain of perspective.

In a real life scenario in a training dataset we may have a large number of features describing the instances. A data analyst should be able to parse a decision tree, see the rule that derived to a specific decision and then draw very accurate conclusions In this paper we will address classification and visualization issues by adding new functionalities and improving the decision tree visualization.

Several classification algorithms have been previously contributed to WEKA but non of them is able to output a data model that is loaded with instances. Based on the previous statement it is clear that there aren't WEKA visualization techniques that are able to present the data in the model in a efficient way and also, there are no available parsing methods ready to implement such functionalities. Traversal of leaves is another task that is missing and it is important because instances from neighbour leaves have a high degree of similarity and share many attributes with similar values.

One aspect that differs at WEKA from other similar software regards its architecture that allows developers to contribute in a productive way. All the work that needs to be done refers to creating a specific folders layout, completing a "description.props" file, adding the ".jar" file to the archive and the build script.

2. RELATED WORK

WEKA is a open source machine learning library that allows developers and researchers to contribute very easily. There are more than twenty years since WEKA had it's first release [9] and there were constant contributions added on it. Not only machine learning algorithms were implemented, for example, in 2005 a text data mining module was developed [20]. An overview of the actual software was made in [8].

Several classifiers were developed and contributed as packages to WEKA. In 2007 a classifier that was build based on a set of sub-samples was developed [14] and compared to C4.5 [15] which have it's implementation called J48 [11] in WEKA. Other classifiers refers the "Alternating Decision Trees Learning Algorithms" [7] which is a generalization of the decision trees, voted decision trees and voted decision stumps. This kind of classifiers are relatively easy to interpret and the rules are usually smaller in size. Classical decision trees, such as c4.5 were expanding nodes in a depth-first order; an improvement came from "Best-first decision trees" [17]which expands nodes in a best-first order. A package with these trees was contributed to WEKA.

Some other contributions refers libraries of algorithms that can be accessed via WEKA. One of them is JCLEC [5] an evolutionary computation framework which has been successfully employed for developing several evolutionary algorithms. Other environment for machine learning and data mining knowledge discovery that was contributed to WEKA is R [10]. This contribution was developed in order to include different sets of tools from both environments available in a single unified system.

Also as related work we must take into consideration some of the last algorithms development. In the last year it is presented a new fast decision tree algorithm [19]. Based on their experiments, the classifier outperforms C5.0 which is the commercial implementation of C4.5.

3. SYSTEM DESIGN

The package is designed to be used both by developers, in their Java applications, and researchers, using the WEKA Explorer. At the moment of writing this paper the package with the Advanced Classifier is still under development, offering more functionalities as a tool for developers than in the explorer view of WEKA.

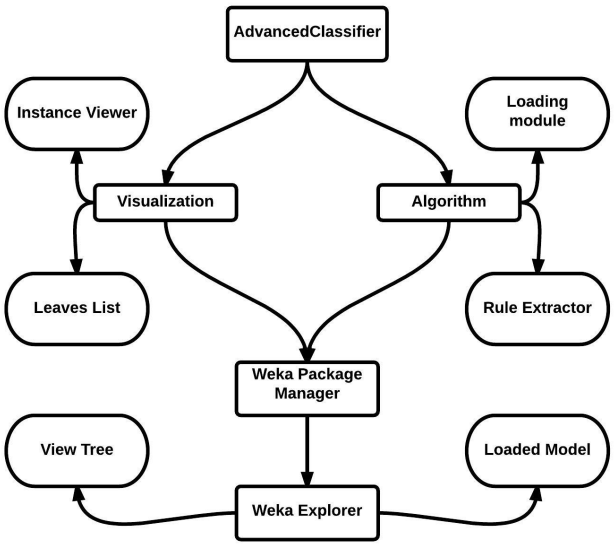

Figure 1: Package Integration in WEKA

In Fig. 1 we present the main design of the algorithm and how it can be used in WEKA. On the top of the figure we have the classifier which can be divided in two main modules: the algorithm and the visualization. As we can see on the next level, both of the modules can be divided further. All the functionalities are then installed in WEKA via the package manager and then, in the explorer, we can perform data analysis tasks using a model loaded with data and it's associated visualization techniques.

3.1 General Architecture

The packages is a zip archive, structured with respect to the WEKA guidelines. That is, it unpacks to the current directory and it contains: the source files, a folder with the required libraries, a build script, a properties file required by WEKA for installing and managing the package, and the actual ".jar" file. A detailed structure of the package is presented below.

```
<current directory>
  +-AdvancedClassifier.jar
  +-Description.props
  +-build_package.xml
  +-src
  |  +-main
  |    +-java
  |      +-resources
  |      |  +-background_node.png
  |      |  +-background_leaf.png
  |      |  +-background_leaf_pressed.png
  |      |  +-font_node.ttf
  |      |  +-font_decision.ttf
  |      +-weka
  |          +-classifiers
  |          |  +-trees
  |          |    +-model
  |          |    |  +-AdvancedClassifierTree.java
  |          |    |  +-AdvancedClassifierTreeBaseNode.java
  |          |    |  +-AdvancedClassifierTreeNode.java
  |          |    |  +-AdvancedClassifierTreeLeaf.java
  |          |    |  +-BaseAttributeValidator.java
  |          |    |  +-NominalAttributeValidator.java
  |          |    |  +-NumericAttributeValidator.java
  |          |    |  +-Constants.java
  |          |    +-AdvancedClassifier.java
  |          |    +-WekaTextfileToXMLTextfile.java
  |          +-gui
  |             +-visualize
  |                +-plugins
  |                   +-AdvancedClassifierTree.java
  |                   +-AdvancedClassifierTreePanel.java
  |                   +-BaseNodeView.java
  |                   +-AdvancedClassifierTreeNodeView.java
  |                   +-AdvancedClassifierTreeLeafView.java
  |                   +-ConnectingLineView.java
  +-lib
    +-weka.jar
    +-simple-xml.jar
    +-rt.jar
```

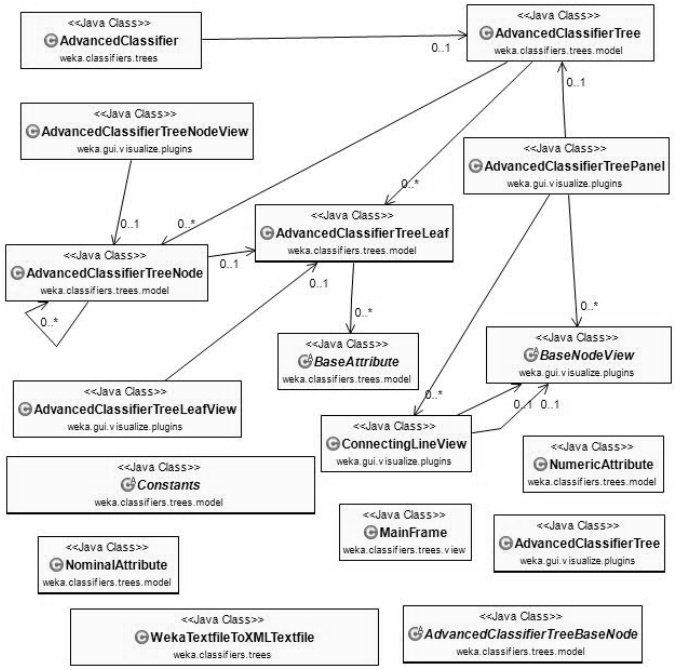

Figure 2: Class Diagram

In Figure 2 is presented the system's class diagram. This diagram includes all the java packages from the project and their relations. As we can see in the above mentioned figure, we have two type of classes: independent classes and composed. Independent classes are gathered from the model part of the Model-View-Controller architecture or just classes that perform one time tasks like "WekaTextFileToXMLTextFile" which is able to generate an XML based on the text file outputted by WEKA. On the other side, the composed classes are dependent on each other and these relations are shared across packages. One important class that is worth to be mentioned is "AdvancedClassifierTreeLeaf.java" in which we store the leaves of our tree along with rules that define the leaf. Discussions about implementation of the packages are more related to the software engineering research area and beyond of the scope of this paper.

3.1.1 Design and Implementation of the Algorithm

The algorithm needs to generate custom rules (dependent on the training dataset) for every leaf of the decision tree. These rules are computed by tracing the path from the root of the tree to the specified leaf. Each decision that leads to a leaf is therefore translated into a rule that encapsulates the name of the attribute and the value on which the decision was made. For each type of attribute defined by WEKA, we need to have a corresponding rule that matches that type. For this purpose an abstract class has been created to act as a base class for any of the custom rules. The name of this class is "BaseAttributeValidator" and exposes the required methods that a superclass needs to implement: a "clone" method required by the workflow of the system and methods that validate if an instance or set of instances have the required values of the attribute targeted by the rule. At the moment, the only implemented rules are the ones that handle "NOMINAL" and "NUMERIC" attribute types.

The rule that validates each nominal attribute is called "NominalAttributeValidator" and receives as parameters the name of the targeted attribute and a string variable representing the accepted value of the attribute. The rule that handles the numeric attributes is called "NumericAttributeValidator" and also receives the name of the attribute and either a particular value or the boundaries of an interval.

In the following paragraphs, we present a brief overview of the algorithm for which we adopt a straightforward approach.

Firstly, the algorithm retrieves instances from the ".arff" file using the methods provided by WEKA. The next step is applying the desired classification process. Currently the only supported classifier is J48, but employing other decision tree classifiers is foreseen as future work. Using the text representation of the outputted model and a predefined set of rules and tags, an XML is then generated. This is an important step during the workflow because the structured XML format allows us to obtain the base model for our decision tree. The deserialization is done using a third-party Java library("Simple XML" [1]).

The model obtained this way contains a list of nodes and leaves with the following significance: each node corresponds to a decision in the tree; the data stored in each object

(node) refers the information about the name of the actual attribute, operator and value on which the decision was made; and the results to which making the decision leads (a list of other nodes or an output leaf). Using this model and the set of attributes provided by WEKA, the set of rules is computed. This step is performed by parsing the model from the first node (i.e., the root) to the last available leaf and gradually composing the set of rules that defines each leaf. The setup of the algorithm is finally completed with the loading of the training dataset into the model.

The classifier and processed data can now be easily handled and different operations can be applied. The method currently implemented include basic per leaf manipulation of instances, i.e. loading new instances into the model and retrieving the part of the dataset contained in each leaf, as well as predecessor and successor computation.

3.1.2 *Visualization Plugin*

For the visualization feature, a custom panel has been designed to hold the components that build up the decision tree and expose the data available in the leaves. The contructor of the panel requires the decision tree model as a parameter, and takes care of adding the corresponding views to the interface. In order to include this functionality in WEKA, a specialized class that implements WEKA's TreeVisualizePlugin interface has been created. After adding the package through the Package Manager and selecting this visualization option, a new JFrame that holds the custom panel is displayed.

```
@RELATION StudentClass

@ATTRIBUTE userid numeric
@ATTRIBUTE nrHours numeric
@ATTRIBUTE avgMark numeric
@ATTRIBUTE present {NO,YES}
@ATTRIBUTE class {low,high}

@DATA
4,17,6.025,NO,low
5,11,7.0,YES,low
7,30,7.375,NO,low
8,10,7.714286,NO,low
9,43,5.1666665,YES,high
10,31,4.5,YES,high
```

Figure 3: Sample from the Dataset

In Figure 3 we present a dataset sample. In order to validate the classifier and it's extra functionalities several tests have been made but for this case study we used three attributes and 788 instances. The feature called "userid" doesn't provide any information gain but can be easily used for instances localization in leaves. The attributes significance is beyond the scope of this paper.

In Figure 4 is presented a screen-shot of the tree generated based on the dataset from figure 3. Each node contains

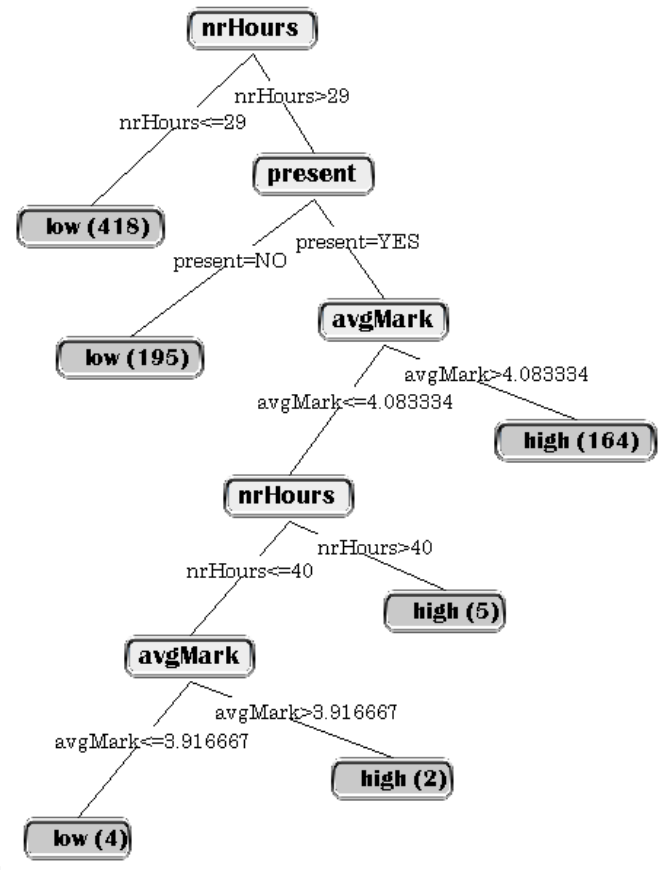

Figure 4: Tree Sample

the name of the attribute, and each decision is printed on top of the connecting line. Surely, each leaf can be clicked, and the set of enclosed instances is displayed. As previously noted, there is still some work to be made to finalize the development of the package, and the visualization tool needs to be included as well. Efforts will have to be made toward providing the means to visualize and handle the successors/predecessors, outliers and other relevant information.

4. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the integration of a data analysis tool in WEKA. This tool is important because brings a new classifier to WEKA that aims to improve the classification procedures. Here, are also presented some implementing procedures and details.

A workflow is also described and all the mechanism that is used to bring new features for the users. One important thing that needs to be mentioned is that the data loading module opens new data analysis opportunities for researchers.

As future work we plan to implement Other types of attributes supported by WEKA like "DATE", "String" and "Relational".

5. REFERENCES

[1] Simple xml. http://simple.sourceforge.net.

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. FernÃąndez, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[3] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. Knime - the konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, Nov. 2009.

[4] R. Campagni, D. Merlini, R. Sprugnoli, and M. C. Verri. Data mining models for student careers. *Expert Systems with Applications*, (0):–, 2015.

[5] A. Cano, J. M. Luna, J. L. Olmo, and S. Ventura. Jclec meets weka! In E. Corchado, M. Kurzynski, and M. Wozniak, editors, *HAIS (1)*, volume 6678 of *Lecture Notes in Computer Science*, pages 388–395. Springer, 2011.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, Nov. 1996.

[7] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 124–133, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[9] G. Holmes, A. Donkin, and I. H. Witten. Weka: a machine learning workbench. pages 357–361, August 1994.

[10] K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: R meets weka. *Computational Statistics*, 24(2):225–232, 2009.

[11] W.-Y. Loh. *Classification and Regression Tree Methods*. John Wiley & Sons, Ltd, 2008.

[12] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 935–940, New York, NY, USA, 2006. ACM.

[13] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in Action*. Manning Publications Co., Greenwich, CT, USA, 2011.

[14] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, 28(4):414–422, 2007.

[15] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[16] C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135 – 146, 2007.

[17] H. Shi. Best-first decision tree learning. Technical report, University of Waikato, 2007.

[18] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[19] S.-G. P. V. Purdila. Fast decision tree algorithm. *Advances in Electrical and Computer Engineering*, 14(1):65–68, 2014.

[20] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA, 1999. ACM.