World Research Journal of Computer Architecture
ISSN: 2278-8514 & E-ISSN: 2278-8522, Volume 1, Issue 1, 2012, pp.-01-05.
Available online at http://www.bioinfo.in/contents.php?id=97

WEKA - OPEN SOURCE TECHNOLOGY, ITS IMPLEMENTATION AND BENIFITS

DESHMUKH J.J. AND TATED R.R.

Department of Computer Science & Engineering, Jawaharlal Darda Institute of Engineering & Technology, Yavatmal- 445001, MS, India.
*Corresponding Author: Email- deshmukhjanhavi65@gmail.com and ruchikatated@gmail.com

Received: February 24, 2012; Accepted: May 03, 2012

Abstract- We've been sharing software since computers were invented. Open source is about sharing source code and keeping it sharable. WEKA (Waikato Environment for Knowledge Analysis) is open source software which consists of a collection of machine learning algorithms for data mining tasks. WEKA implements algorithms for data preprocessing, classification, regression, clustering and association rules; it also includes visualization tools. This paper mainly deals with what is open source, how open source software (OSS) is developed, difference between OSS and closed source software, what is the need for it to emerge, what are its advantages and WEKA. Paper also states the effect of this type of system on the other modes of software's such as closed or proprietary software.
Keywords- machine learning software, open source software, source code, closed source software, WEKA, data preprocessing, classification, regression, clustering and association rules.

Citation: Deshmukh J.J. and Tated R.R. (2012) Weka - Open Source Technology, Its Implementation and Benefits. World Research Journal of Computer Architecture, ISSN: 2278-8514 & E-ISSN: 2278-8522, Volume 1, Issue 1, pp.-01-05.

Introduction

Open source refers to a program in which the *source code* (the form of the program when a programmer writes a program in a particular programming language) is available to the general public for use and /or modifications from its original design free of charge, i.e. open. Open source code is typically created as a collaborative effort in which programmer improve upon the code and share the changes within the community. Open source sprouted in the technological community as a response to *proprietary software* (made and sold only by owner) owned by corporations.

Need for Open Source Software

1. Reduce dependency on closed source vendors.
2. Your annual budget does not keep up with increases in software maintenance costs and increased costs of employee health care.
3. More access to tools.
4. Try before you buy.
5. Great support and a 24/7 online community that responds quickly.
6. Access to source code and the ability to customize if you desire.
7. More secure than most closed source vendors.
8. Bug fixes are implemented faster then closed source vendor [4].

Benefits of Open Source Software

The following are the advantages that open source code offers over closed source.

Bug-Fixing

All software contains bugs. The people developing the software will have spotted and dealt with bugs. When a bug is spotted in proprietary software, the only people who can fix it are the original developers, as only they have access to the source code. Open source software is different. As users can access and change the code, bugs tend to be more visible and more rapidly corrected. One of the slogans of the open source movement is that 'Given enough eyeballs, all bugs are shallow' (Eric Raymond, *The Cathedral and the Bazaar*) [6].

Security
Having access to the source code allows the user of that software to choose the approach to security that they want. It allows you to take ownership of your own security. It is possible to decide on your own security priorities and to allocate resources accordingly.
Access to source code makes it easier to detect security flaws in software, whether you are looking to fix them or exploit them. In practice, the skills and time required to find security flaws, work out how they can be exploited, and then initiate an attack, are more specialized than the *mundane debugging* (common) skills required closing exploits [6].

Customization
Closed source applications can only be customized or adapted within the scope provided by the original vendor but never outside its boundaries. Open source applications may be customized by anyone with the requisite skill. Thus, open source software can be readily adapted to meet specific user needs. Even if you cannot program yourself, if you would like something added or customized you can generally pay an appropriately skilled software developer to do it for you. The introduction of competition into the market for customizations as observed in the bug-fixing section above, forces suppliers to offer high quality at a competitive price.
However, when modifying open source code it is good practice to ensure that, wherever possible, changes are contributed back upstream to the main project. Failure to do this can result in unnecessary complexities when upgrading to newer versions of the software [6].

Translation
With access to the source code it is easy to translate the language of the software interface. Closed source commercial software vendors are unwilling to translate their products into less widely spoken languages, as the market for them would be too small to guarantee profit [6].

Avoiding Lock-in
Organizations are said to be *'locked-in'* to software products when the costs of switching to alternatives are prohibitively high. Proprietary software vendors can 'lock' users in to their products by ensuring that they are not readily compatible with potential rivals. Vendors may then increase the price of product upgrades or support without too great a risk of losing existing customers.
As there is no incentive to use non-standard formats to inhibit compatibility, open source software tends to use open standard formats and there is little danger of being 'locked-in' by a vendor. Even when non-standard formats are used in open-source code, it is always possible to document them from the source code. On the contrary, closed formats used by proprietary software need to be reverse-engineered, a burdensome and expensive process that may need to be repeated if the format is subsequently changed [6].

Mitigation of Vendor Collapse or Product Discontinuation
Commercial software vendors go bust or get bought up from time to time. When this happens, there is no guarantee that their software products will continue to be available, supported, or updated. This can result in users needing to switch products, which can be very expensive and difficult, especially if they were heavily 'locked-in' to their current product.
With open source software, this danger is greatly reduced. As the source code is not 'owned' in the same way that proprietary source code is, it may be picked up and developed by anyone with an interest in a product's survival [6].

Learning from Examples
Open source code provides an excellent resource from which to learn, and open source projects provide a practical environment in which to test your skills. Just watching the development process can provide an education in itself. If you choose to submit code to an open source project, it will generally be checked and commented on by experienced programmers. Once you have convinced the project community that your code is of appropriate quality, you may be granted full committer rights yourself [6].

Being Part of a Community
By adopting open source software you become part of a community of users and developers who have an interest in working together to support each other and improve the software. The extent to which you engage with this community is up to you, but you may obtain the intangible benefits of goodwill if you do [6].

Cost
Open source programs can be obtained at no cost or at a very low cost. This is often an important issue for individuals and in many cases this has been the main reason for an individual adopting a particular open source solution over a closed source alternative. Other costs may arise: training, consulting, maintenance [6].

Examples of Open Source Software
- WEKA (Waikato Environment for Knowledge Analysis)
- Apache HTTP Server (web server)
- Blender (3D graphics and animation package)
- DSpace (digital repository)
- EPrints (digital repository)
- The GIMP (image editor)
- GNOME (Linux desktop environment)
- GNU Compiler Collection (GCC, a suite of compilation tools for C, C++, etc)
- KDE (Linux desktop environment)
- LORLS (reading lists management system)
- Mailman (mailing list manager)
- Moodle (virtual learning system)
- Mozilla (web browser and email client)
- Firefox (web browser based on Mozilla)
- Thunderbird (mail client based on Mozilla code)
- MySQL (database)
- OpenOffice.org (office suite, including word processor, spreadsheet, and presentation software)
- PHP (web development)
- Perl (programming/scripting language)
- Plone (content management system)
- PostgreSQL (database)
- Python (programming/scripting language)
- Sakai (learning management system)
- Samba (files and print server)

- SSL-Explorer: Community Edition (browser-based SSL VPN solution)
- TeX (typesetting language)

Introduction to WEKA

WEKA is a data mining system developed by the University of Waikato in New Zealand that implements data mining algorithms using the JAVA language. WEKA is a state of-the-art facility for developing machine learning (ML) techniques and their application to real-world data mining problems. It is a collection of machine learning algorithms for data mining tasks. The algorithms are applied directly to a dataset. WEKA implements algorithms for data preprocessing, classification, regression, clustering and association rules; It also includes visualization tools. The new machine learning schemes can also be developed with this package. WEKA is open source software issued under General Public License [5].

Features of WEKA

Data preprocessing

Data preprocessing as well as a native file format ARFF (Attribute Relation File Format), WEKA supports various other formats (for instance CSV (Comma Separated Values (text file)), Mat lab ASCII files), and database connectivity through JDBC. Data can be filtered by a large number of methods (over 75), ranging from removing particular attributes to advanced operations such as principal component analysis. The XRFF (Xml attribute Relation File Format) is a representing the data in a format that can store comments, attribute and instance weights.

A complete description of the ARFF file Format can be found here

% This is a toy example, the UCI weather dataset.

% Any relation to real weather is purely coincidental.

Comment lines at the beginning of the dataset should give an indication of its source, context and meaning.

@relation golfWeatherMichigan_1988/02/10_14days

Here we state the internal name of the dataset. Try to be as comprehensive as possible.

@attribute outlook {sunny, overcast rainy}

@attribute windy {TRUE, FALSE}

Here we define two nominal attributes, outlook and windy. The former has three values: sunny, overcast and rainy; the latter two: TRUE and FALSE. Nominal

Values with special characters, commas or spaces are enclosed in 'single quotes'.

@attribute temperature real

@attribute humidity real

These lines define two numeric attributes. Instead of real, integer or numeric can also be used. While double floating point values are stored internally, only seven decimal digits are usually processed.

@attribute play {yes, no}

The last attribute is the default target or class variable used for prediction. In our case it is a nominal attribute with two values, making this a binary classification problem [2,3].

Classification

One of WEKA's drawing cards is the more than 100 classification methods it contains. Classifiers are divided into "Bayesian" methods (Naive Bayes, Bayesian nets, etc.), lazy methods (nearest neighbor and variants), rule-based methods (decision tables, OneR, RIPPER), tree learners (C4.5, Naive Bayes trees, M5), function-based learners (linear regression, SVMs, Gaussian processes), and miscellaneous methods. Furthermore, WEKA includes Meta-classifiers like bagging, boosting, stacking; multiple instance classifiers; and interfaces for classifiers implemented in Groovy and Jython.

Selecting a Classifier

At the top of the classify section is the Classifier box. This box has a text field that gives the name of the currently selected classifier, and its options. Clicking on the text box with the left mouse button brings up a GenericObjectEditor dialog box, just the same as for filters, which you can use to configure the options of the current classifier. With a right click (or Alt+Shift+left click) you can once again copy the setup string to the clipboard or display the properties in a GenericObjectEditor dialog box. The Choose button allows you to choose one of the classifiers that are available in WEKA [2,3].

Clustering

Unsupervised learning is supported by several clustering schemes, including EM based mixture models, *k*-means, and various hierarchical clustering algorithms. Though not as many methods are available as for classification. Most of the classic algorithms are included. Selecting and configuring objects. Clicking on the clustering scheme listed in the Clusterer box at the top of the window brings up a GenericObjectEditor dialog with which to choose a new clustering scheme.

Cluster Modes

The Cluster mode box is used to choose what to cluster and how to evaluate the results. The first three options are the same as for classification: Use training set, Supplied test set and Percentage split. Now the data is assigned to clusters instead of trying to predict a specific class. The fourth mode, Classes to clusters evaluation, compares how well the chosen clusters match up with a pre-assigned class in the data. The drop-down box below this option selects the class, just as in the Classify panel. An additional option in the Cluster mode box, the Store clusters for visualization tick box, determines whether or not it will be possible to visualize the clusters once training is complete. When dealing with datasets that are so large that memory becomes a problem it may be helpful to disable this option [2,3].

Attribute Selection

The set of attributes used is essential for classification performance. Various selection criteria and search methods are available. In box titled Attributes. There are four buttons, and beneath them is a list of the attributes in the current relation. The list has three columns:

1. No. - A number that identifies the attribute in the order they are specified in the data file.
2. Selection tick boxes. These allow you select which attributes are present in the relation.

3. Name- The name of the attribute, as it was declared in the data file.

When you click on different rows in the list of attributes, the fields change in the box to the right titled Selected attribute. This box displays the characteristics of the currently highlighted attribute in the list:

1. Name- The name of the attribute, the same as that given in the attribute list.
2. Type- The type of attribute, most commonly Nominal or Numeric.
3. Missing- The number (and percentage) of instances in the data for which this attribute is missing (unspecified).
4. Distinct- The number of different values that the data contains for his attribute.
5. Unique- The number (and percentage) of instances in the data having a value for this attribute that no other instances have.

Below these statistics is a list showing more information about the values stored in this attribute, which differ depending on its type. If the attribute is nominal, the list consists of each possible value for the attribute along with the number of instances that have that value. If the attribute is numeric, the list gives four statistics describing the distribution of values in the data—the minimum, maximum, mean and standard deviation. And below these statistics there is a colored histogram, colour-coded according to the attribute chosen as the Class using the box above the histogram. (This box will bring up a drop-down list of available selections when clicked.) Note that only nominal Class attributes will result in a colour-coding. Finally, after pressing the Visualize All button, histograms for all the attributes in the data are shown in a separate window. Returning to the attribute list, to begin with all the tick boxes are unticked. They can be toggled on/off by clicking on them individually. The four buttons above can also be used to change the selection [2,3].

Association

Setting Up

This panel contains schemes for learning association rules, and the learners are chosen and configured in the same way as the clusterers, filters, and classifiers in the other panels.

Learning Associations

Once appropriate parameters for the association rule learner have been set, click the Start button. When complete, right-clicking on an entry in the result list allows the results to be viewed or saved [2].

Data Visualization

Data can be inspected visually by plotting attribute values against the class, or against other attribute values. Classifier output can be compared to training data in order to detect outliers and observe classifier characteristics and decision boundaries. For specific methods there are specialized tools for visualization, such as a tree viewer for any method that produces classification trees, a Bayes network viewer with automatic layout, and a dendrogram viewer for hierarchical clustering. WEKA also includes support for association rule mining, comparing classifiers, dataset generation, facilities for annotated documentation generation for source code, distribution estimation, and data conversion. WEKA's visualization section allows you to visualize 2D plots of the current relation.

The Scatter Plot Matrix

When you select the Visualize panel, it shows a scatter plot matrix for all the attributes, colour coded according to the currently selected class. It is possible to change the size of each individual 2D plot and the point size, and to randomly jitter the data (to uncover obscured points). It also possible to change the attribute used to colour the plots, to select only a subset of attributes for inclusion in the scatter plot matrix, and to sub sample the data. Note that changes will only come into effect once the Update button has been pressed [2,3].

Application with WEKA

In most data mining applications the machine learning component is just a small part of a far larger software system. To accommodate this, it is possible to access the programs in Weka from inside one's own code. This allows the machine learning sub problem to be solved with a minimum of additional programming.

For example, Figure shows a Weka applet written to test the usability of machine learning techniques in the objective measurement of mushroom quality. Image processing a picture of a mushroom cap (at left in Figure) provides data for the machine learning scheme to differentiate between A, B and C grade mushrooms [1].

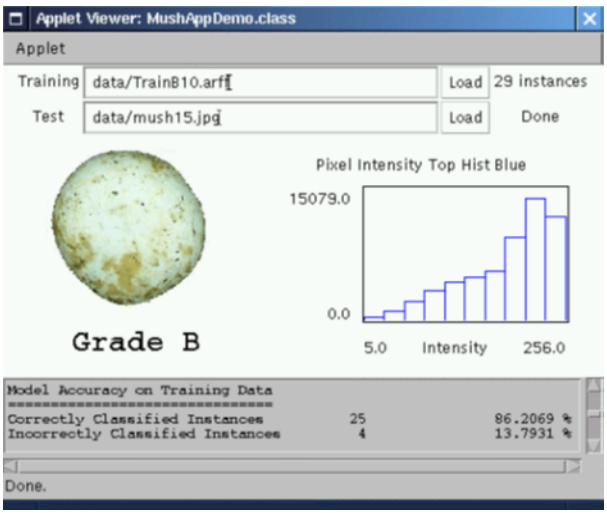

Fig. 1- Mushroom grading applet

Advantages

- It is used in the machine learning and data mining community as an educational tool for teaching both applications and technical internals of machine learning algorithms.
- It is also used as a research tool for developing and empirically comparing new techniques.
- It is applied academic fields, and in commercial settings.
- It is free and open-source software.

Conclusion

From the above discussion we can conclude that Open-source software is often effectively publicly owned and can be used for key public interest activities, vs. private ownership, which prevents

this from happening.

At present to access and use any information is dependent on private companies which design software to benefit their share-holders. By using open source software, we can modify the source code and improve as well as increase the functionality of that software.

As the technology of machine learning continues to develop and mature, learning algorithms need to be brought to the desktops of people who work with data and understand the application domain from which it arises. It is necessary to get the algorithms out of the laboratory and into the work environment of those who can use them. Weka is a significant step in the transfer of machine learning technology into the workplace.

References

[1] Kusabs N., Bollen F., Trigg L., Holmes G. and Inglis S. (1998) *Proc New Zealand Institute of Agricultural Science and the New Zealand Society for Horticultural Science Annual Convention*, Hawke's Bay, New Zealand, 51.

[2] Remco R. Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald and David Scuse (2008) *WEKA Manual for Version* 3-6-0.

[3] Remco R. Bouckaert, Eibe Frank, Mark A. Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (2010) *Journal of Machine Learning Research* 11, 2533-2541.

[4] *http://it.toolbox.com/blogs/madgreek/10-reasons-why-you-need-an-open-source-strategy-18891*.

[5] *http://www.cs.waikato.ac.nz/ml/weka*.

[6] *http://www.webopidea.com*.