CHAPTER

Introduction to Weka 10

Experience shows that no single machine learning scheme is appropriate to all data mining problems. The universal learner is an idealistic fantasy. As we have emphasized throughout this book, real datasets vary, and to obtain accurate models the bias of the learning algorithm must match the structure of the domain. Data mining is an experimental science.

The Weka workbench is a collection of state-of-the-art machine learning algorithms and data preprocessing tools. It includes virtually all the algorithms described in this book. It is designed so that you can quickly try out existing methods on new datasets in flexible ways. It provides extensive support for the whole process of experimental data mining, including preparing the input data, evaluating learning schemes statistically, and visualizing the input data and the result of learning. As well as a variety of learning algorithms, it includes a wide range of preprocessing tools. This diverse and comprehensive toolkit is accessed through a common interface so that its users can compare different methods and identify those that are most appropriate for the problem at hand.

Weka was developed at the University of Waikato in New Zealand; the name stands for *Waikato Environment for Knowledge Analysis*. (Outside the university, the weka, pronounced to rhyme with *Mecca*, is a flightless bird with an inquisitive nature found only on the islands of New Zealand.) The system is written in Java and distributed under the terms of the GNU General Public License. It runs on almost any platform and has been tested under Linux, Windows, and Macintosh operating systems—and even on a personal digital assistant. It provides a uniform interface to many different learning algorithms, along with methods for pre- and postprocessing and for evaluating the result of learning schemes on any given dataset.

10.1 WHAT'S IN WEKA?

Weka provides implementations of learning algorithms that you can easily apply to your dataset. It also includes a variety of tools for transforming datasets, such as the algorithms for discretization described in Chapter 7. You can preprocess a dataset, feed it into a learning scheme, and analyze the resulting classifier and its performance—all without writing any program code at all.

The workbench includes methods for the main data mining problems: regression, classification, clustering, association rule mining, and attribute selection. Getting to know the data is an integral part of the work, and many data visualization facilities and data preprocessing tools are provided. All algorithms take their input in the form of a single relational table in the ARFF format described in Section 2.4, which can be read from a file or generated by a database query.

One way of using Weka is to apply a learning method to a dataset and analyze its output to learn more about the data. Another is to use learned models to generate predictions on new instances. A third is to apply several different learners and compare their performance in order to choose one for prediction. In the interactive Weka interface, you select the learning method you want from a menu. Many methods have tunable parameters, which you access through a property sheet or *object editor*. A common evaluation module is used to measure the performance of all classifiers.

Implementations of actual learning schemes are the most valuable resource that Weka provides. But tools for preprocessing the data, called *filters*, come a close second. Like classifiers, you select filters from a menu and tailor them to your requirements. We will show how different filters can be used, list the filtering algorithms, and describe their parameters. Weka also includes implementations of algorithms for learning association rules, clustering data for which no class value is specified, and selecting relevant attributes in the data, which we describe briefly.

10.2 HOW DO YOU USE IT?

The easiest way to use Weka is through a graphical user interface called *Explorer*. This gives access to all of its facilities using menu selection and form filling. For example, you can quickly read in a dataset from an ARFF file (or spreadsheet) and build a decision tree from it. But learning decision trees is just the beginning: There are many other algorithms to explore. The Explorer interface helps you do just that. It guides you by presenting choices as menus, by forcing you to work in an appropriate order by graying out options until they are applicable, and by presenting options as forms to be filled out. Helpful *tool tips* pop up as the mouse passes over items on the screen to explain what they do. Sensible default values ensure that you can get results with a minimum of effort—but you will have to think about what you are doing to understand what the results mean.

There are two other graphical user interfaces to Weka. The *Knowledge Flow* interface allows you to design configurations for streamed data processing. A fundamental disadvantage of the Explorer interface is that it holds everything in main memory—when you open a dataset, it immediately loads it all in. This means that the Explorer can only be applied to small- to medium-size problems. However, Weka contains some incremental algorithms that can be used to process very large datasets. The Knowledge Flow interface lets you drag boxes representing learning algorithms and data sources around the screen and join them together into the configuration you

want. It enables you to specify a data stream by connecting components representing data sources, preprocessing tools, learning algorithms, evaluation methods, and visualization modules. If the filters and learning algorithms are capable of incremental learning, data will be loaded and processed incrementally.

Weka's third interface, *Experimenter*, is designed to help you answer a basic practical question when applying classification and regression techniques: Which methods and parameter values work best for the given problem? There is usually no way to answer this question a priori, and one reason we developed the workbench was to provide an environment that enables Weka users to compare a variety of learning techniques. This can be done interactively using the Explorer interface. However, the Experimenter interface allows you to automate the process by making it easy to run classifiers and filters with different parameter settings on a corpus of datasets, to collect performance statistics, and to perform significance tests. Advanced users can employ Experimenter to distribute the computing load across multiple machines using Java remote method invocation (RMI). In this way, you can set up large-scale statistical experiments and leave them to run.

Behind these interactive interfaces lies the basic functionality of Weka. This can be accessed in raw form by entering textual commands, which gives access to all features of the system. When you fire up Weka, you have to choose among four different user interfaces: the Explorer, the Knowledge Flow, the Experimenter, and command-line interfaces. We describe them in turn in the next chapters. Most people choose Explorer, at least initially.

10.3 WHAT ELSE CAN YOU DO?

An important resource when working with Weka is the online documentation, which has been automatically generated from the source code and concisely reflects its structure. We will explain how to use this documentation. We will also identify Weka's major building blocks, highlighting which parts contain supervised learning methods, which contain tools for data preprocessing, and which contain methods for other learning schemes. The online documentation gives the only complete list of available algorithms because Weka is continually growing and—being generated automatically from the source code—the online documentation is always up to date. Moreover, it becomes essential if you want to proceed to the next level and access the library from your own Java programs or write and test learning schemes of your own.

In most data mining applications, the machine learning component is just a small part of a far larger software system. If you intend to write a data mining application, you will want to access the programs in Weka from inside your own code. By doing so, you can solve the machine learning subproblem of your application with a minimum of additional programming. We show how to do that by presenting an example of a simple data mining application in Java. This will enable you to become

familiar with the basic data structures in Weka, representing instances, classifiers, and filters.

If you intend to become an expert in machine learning algorithms (or, indeed, if you already are one), you'll probably want to implement your own algorithms without having to address such mundane details as reading the data from a file, implementing filtering algorithms, or providing code to evaluate the results. If so, we have good news for you: Weka already includes all this. To make full use of it, you must become acquainted with the basic data structures. To help you reach this point, we will describe these structures in more detail and explain an illustrative implementation of a classifier in Chapter 16.

10.4 HOW DO YOU GET IT?

Weka is available from *www.cs.waikato.ac.nz/ml/weka*. You can download either a platform-specific installer or an executable Java jar file that you run in the usual way if Java is installed. We recommend that you download and install it now, and follow through the examples in the upcoming chapters.