**How to Use Weka**

1. **What is Weka?**

Weka is a collection of machine learning algorithms for solving real-world data mining problems, which is developed in the University of Waikato, New Zealand. It is written in Java and runs on almost any platform. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU public license. More information can be found from:

<http://www.cs.waikato.ac.nz/ml/weka>

1. **How to Install Weka at home?**

* Create a subdirectory *weka* under the C (or D) drive
* Download the weka-3-6-13.exe (windows version) file from

<http://www.cs.waikato.ac.nz/ml/weka/downloading.html>

select the version without the Java VM, if you have Java installed.

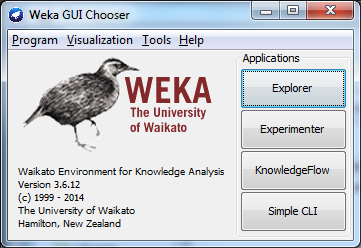
* Run it and the Weka will be installed automatically.
* A directory weka-3-6 and three subdirectories will be generated:
* Data includes some sample arff format data files
* Doc includes weka packages
* Changelogs includes the evolution information about Weka
* Also there is a weka manual (wekamanual.pdf) available

1. **How to Run Weka**

To start weka,

* at home, follow
* Double click the weka.jar or
* Start 🡪 All Programs 🡪 WEKA 🡪weka-3-6

This will start a small GUI (GUIChooser) which includes four buttons: Explorer, Experimenter, KnowledgeFlow and SimpleCLI. SimpleCLI just acts like a simple command shell and Explorer acts like a window application. We are not going to use SimpleCLI, the Experimenter and the KnowledgeFlow in this module. The GUI looks like:



For details of using Explorer, please see WEKA Explorer User Guide for Version 3-5-8 by R. Kirkby, E. Frank and P. Reutemann. This file can be found at <http://www.cs.waikato.ac.nz/ml/weka/documentation.html>. The guide provides a brief introduction to the Explore. The User Manual provides details of the software, including a user guide for the Explore.

1. **Data format**

Datasets in Weka have to be formatted according to the arff format. Examples of arff files can be found in the ./Weka3-6/data directory. What follows is a short description of the file format.

A dataset has to start with a declaration of its name:

@relation name

followed by a list of all the attributes in the dataset (including the class attribute). These declarations have the form

@attribute attribute\_name specification

If an attribute is nominal, specification contains a list of the possible attribute values in curly brackets:

@attribute nominal\_attribute {first\_value, second\_value, third\_value, …}

If an attribute is numeric, specification is replaced by the keyword real: (Integer values are treated as real numbers in Weka.)

@attribute numeric\_attribute real

In addition to these two types of attributes, there also exists a string attribute type. This attribute provides the possibility to store a comment or ID field for each of the instances in a dataset:

@attribute string\_attribute {‘all paid’, ‘no credit’, ‘existing paid’}

After the attribute declarations, the actual data is introduced by a

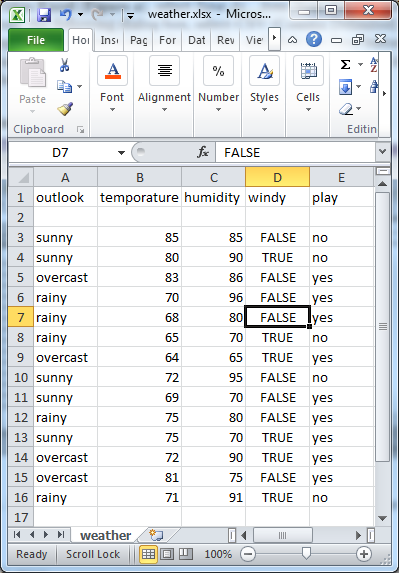
@data

tag, which is followed by a list of all the instances. The instances are listed in comma-separated format, with a question mark representing a missing value. Comments are lines starting with %

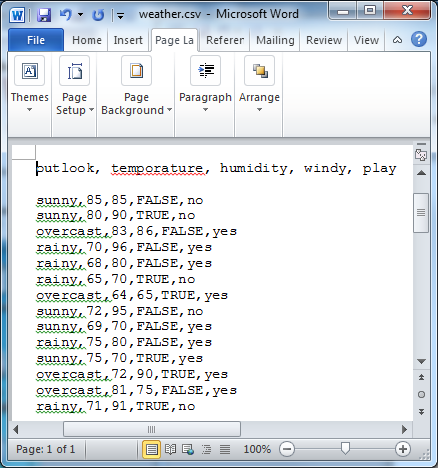
**Data preparation**

Suppose your data is stored in a Microsoft Excel spreadsheet, e.g. **Figure a**. The following method shows how to convert the data from this format to the arff format.

1. Within **Excel** (**Figure a**), select *Save As*…item from the *File* pull-down menu.
2. From the following dialog box, select CSV (Comma Delimited) from the file type popup menu, enter a name for the file, and click the *Save* button. (A message will warn you that this will only save the active sheet: just ignore it by clicking OK.) Then exit.



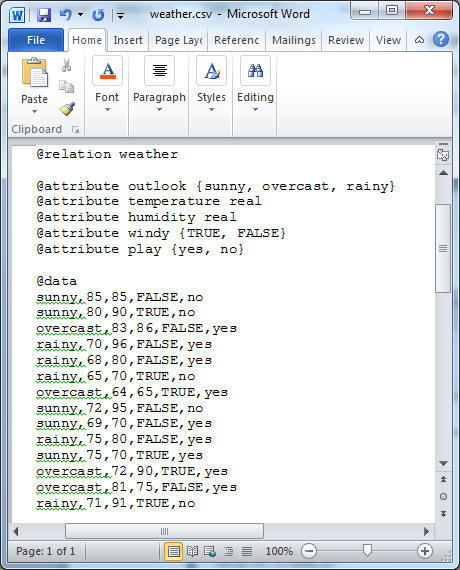
**Figure a**



**Figure b**

1. Load this file into Microsoft Word. The screen will look like **Figure b**.
2. The rows of the original spreadsheet have been converted into lines of texts, and the elements are separated from each other by commas. All you have to do is convert the first line, which holds the attribute names, into the header structure that makes up the beginning of the arff file.

**Figure c** shows the result. The dataset's name is introduced by a @relation tag, and the names, types, and values of each attribute are defined by @attribute tags. The data section of the arff file begins with a @data tag. Once the structure of your dataset matches Figure c, you should save it as a text file:



**Figure c**

* Choose *Save As*… from the *File* menu
* specify *Text Only* as the file type by using the corresponding popup menu
* Enter a file name, and press *Save* button
* Then rename the file's extension from .txt to .arff, e.g., weather.arff.

Note: You can load the .csv file to weak. But the attribute names will be arbitrary.

1. **Meaning of the Output**

The following is the output by using the J48 algorithm (C4.8 decision tree learner) on data weather.arff:

+++++++++++++++++++

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: weather

Instances: 14

Attributes: 5

outlook

temperature

humidity

windy

play

Test mode: evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree

------------------

outlook = sunny

| humidity <= 75: yes (2.0)

| humidity > 75: no (3.0)

outlook = overcast: yes (4.0)

outlook = rainy

| windy = TRUE: no (2.0)

| windy = FALSE: yes (3.0)

Number of Leaves : 5

Size of the tree : 8

Time taken to build model: 0 seconds

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances 14 100 %

Incorrectly Classified Instances 0 0 %

Kappa statistic 1

Mean absolute error 0

Root mean squared error 0

Relative absolute error 0 %

Root relative squared error 0 %

Total Number of Instances 14

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure Class

1 0 1 1 1 yes

1 0 1 1 1 no

=== Confusion Matrix ===

a b <-- classified as

9 0 | a = yes

0 5 | b = no

+++++++++++++++++++++

The first part is a brief description of the run information. The second part is a pruned decision tree in textual form. As you can see the first split is on the *outlook* attribute, and then, at the second level, the splits are on *humidity* and *windy* respectively. In the tree structure, a colon introduces the class label that has been assigned to a particular leaf, followed by the number of instances that reach that leaf, expressed as a decimal number because of the way the algorithm uses fractional instances to handle missing values. For instance, the number 4.0 following outlook = overcast: yes means that there are 4 instances reach this leaf. Below the tree structure, the number of leaves is printed, then the total number of notes in the tree (size of the tree).

**Note**: Sometimes there are more than one number appearing in the bracket. For example, if we have a pair of numbers (67.0/15.0) following the outlook = overcast: yes, it means that there are 67 instances reach this leaf, where 15 out of 67 are incorrectly classified under this leaf.

The second part of the output gives estimates of the tree's predictive performance, generated by Weka's evaluation module. The first set of measurements is derived from the training data. In this case, all fourteen training instances have been classified correctly, and none were left unclassified. An instance can be left unclassified if the learning scheme refrains from assigning any class label to it, in which case the number of unclassified instances will be reported in the output.

In addition to the classification errors, the evaluation module also outputs measurements derived from the class probabilities assigned by the tree. More specifically, it outputs the mean absolute error and the root mean-squared error of the probability estimates. The root mean-squared error is the square root of the average quadratic loss. The mean absolute error is calculated in a similar way by using the absolute instead of the squared difference. In this example, both figures are 0 because the output probabilities for the tree are either 0 or 1, due to the fact that all leaves are pure and all training instances are classified correctly.

The summary of the results from the training data ends with a confusion matrix, showing how many instances of each class have been assigned to each class. Elements show the number of the examples whose actual class is the row and whose predicted class is the column. In this example, only the diagonal elements of the matrix are non-zero because all instances are classified correctly. It shows that 9 examples have actual class “yes” and all of them are predicted as “yes” as well.