Advances and Applications in Mathematical Sciences
Volume 19, Issue 6, April 2020, Pages 451-469
© 2020 Mili Publications

EXPLORING DATA MINING TOOL - WEKA AND USING WEKA TO BUILD AND EVALUATE PREDICTIVE MODELS

KANWAL PREET SINGH ATTWAL and AMARDEEP SINGH DHIMAN

Department of Computer Science and Engineering,
Punjabi University
Patiala, India
E-mail: kanwalp78@yahoo.com
amardeep_dhiman@yahoo.com

Abstract

WEKA is a Java based software suite which implements a large number of machine learning algorithms. It can be used for performing different Data Mining tasks such as Data Preprocessing, Classification, Clustering, Association Rule mining and Visualization. This paper is an attempt to acquaint the researcher with the WEKA interface. The study explores the different file formats supported by WEKA and also discuss the methods of converting excel files to WEKA supported file formats. The paper also illustrates how to use WEKA filters to divide the dataset into two categories – the training dataset and the test dataset. The training dataset is then used to build a predictive model and the test dataset is used to evaluate the performance of the model. The study also shows how the model can be used for the prediction of the class of those tuples/objects, the class label of which is not known.

1. Introduction

WEKA which stands for "Waikato Environment for Knowledge Analysis" is a Java based open-source Data Mining Tool developed by the University of Waikato. It is an amalgamation of Machine Learning Algorithms and Data Preprocessing Tools. The implementations of Machine Learning Algorithms are provided by the WEKA, which are easily applicable to the dataset. Besides, various tools for transforming datasets – as the algorithms for discretisation and sampling – are also available in WEKA. A dataset can be preprocessed, fed into a learning scheme, and the resulting classifier as well

2010 Mathematics Subject Classification: 68P15.
Keywords: WEKA tool, Data Mining, Predictive Model, Classifier, ARFF file format, Model Evaluation.
Received 30 October 2019; Accepted 22 December 2019

as its performance can be analysed. WEKA provides both – GUI and a Command Line Interface to run these algorithms on your dataset. The user does not have to write any code. WEKA requires Java installed on the computer system. The latest stable version of WEKA –3.8–can be downloaded from the link - http://www.cs.waikato.ac.nz/ml/weka/downloading.html. The WEKA workbench includes implementation of algorithms for performing the main data mining tasks: Regression, Classification, Clustering, Association Rule Mining and Attribute Selection (Witten, Frank, and Hall, 2011). The following table summarizes the different Data Mining Tasks that can be performed in WEKA.

Table 1. Data Mining Tasks available in WEKA.

Data Mining Task	Description	Examples
Pre-Processing of Data	Preparing a dataset for analysis	Discretizing, Nominal to Binary
Clustering	Identify groups (i.e., clusters) of similar observations	K-Means
Classification	Examining the features of a newly presented object and assigning a predefined class to it	Bayes Net, KNN, Decision Tree, Neural Networks, Perceptron, SVM
Regression	Predicting numeric values for a particular attribute of a newly presented object	Linear Regression, Isotonic Regression
Association rule mining	Finding relationships among values of different attributes in a database	Apriori Algorithm, Predictive Accuracy
Visualization	Visually represent data mining results	Cluster assignments, ROC curves

The modular, object-oriented architectural design of WEKA workbench permits an easy addition of new classifiers, filters, clustering and other algorithms. A set of abstract Java classes have been designed and placed in a corresponding top-level package. This is done for every major component.

All classifiers are included in subpackages of the top level “classifiers” package and extend a “Classifier” which is a common base class. The Classifier class recommends a public interface and a set of rules. According to the functionality or the purpose, components are grouped together by sub-packages. For instance, filters are first divided into supervised and unsupervised filters, and further on the basis of their operations i.e. either it is attribute based or instance based. Classifiers are arranged in accordance with the general type of learning algorithm, hence there are subpackages for Bayesian methods, tree inducers, rule learners, etc. (Maimon and Rokach, 2010).

“Core” which are supporting classes in a top level package is a representation of attributes and instances. It is one of the many common utility methods. The classes and data structures in this package read data sets. All the components depend on these supporting classes. The core package includes additional interfaces which may be implemented by the components in order to support extra functionality.

When WEKA has been installed, weka-3-8 (or weka-version) folder is found in Program Files, containing a weka.jar file. WEKA stores pre-written packages, classes, and interfaces with their respective methods, fields and constructors in the form of Java API. This API is stored in weka.jar file.

2. Supported File Formats

WEKA does not accept files in normal xlsor xlsx format. ARFF – “Attribute-Relation File Format” is the default file format of WEKA, although it also accepts files in comma-separated value (CSV) format, C4.5 format, etc. ARFF file is an ASCII text file; it gives a list of instances that share a set of attributes(Bouckaert, et al., 2016).The ARFF files have two distinct sections – the Header Section and the Data Section. The details of the relation, the attributes, and their types are contained in the reader of the ARFF file and the Data Section contains the relevant data. Comments can be added by using percent sign (%),the dataset’s name by using the @relation tag and the attribute information by using @attributetag. The first line in the ARFF file is the relation name. The format is:

@relation <relation-name>

where<relation-name> is a string. If the name contains spaces, the string should be quoted.

The format for the @attribute statement is:

@attribute <attribute-name><datatype>

Weka supports four types of data:

- Numeric
- Nominal
- string
- date

Numeric attributes may either be real or integer numbers.

Nominal attributes are used when the value of an attribute is chosen from a pre-defined list. A nominal attribute is defined by providing a "nominal-specification" listing the possible values as: <nominal-name1>, <nominal-name2>, <nominal-name3>, ……

For Iris dataset, the class value is defined as follows:

@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

Values that contain spaces must be quoted.

String is used to create attributes that contain arbitrary text values.

Date attribute is used to store dates in different formats. The declarations take the following form:

@attribute <name> date [<date-format>]

Here<name> is the name for the attribute; <date-format> is an optional string which specifies the way of parsing and printing the date values. The default dates should be specified in the data section as the corresponding string representations of the date/time.

The ARFF Data Section of the file contains the data declaration line and the actual instance lines. A detailed description of ARFF files can be obtained from the Data section of WEKA documentation at the download of WEKA.

2.1. Conversion from xls/xlsv to csv format

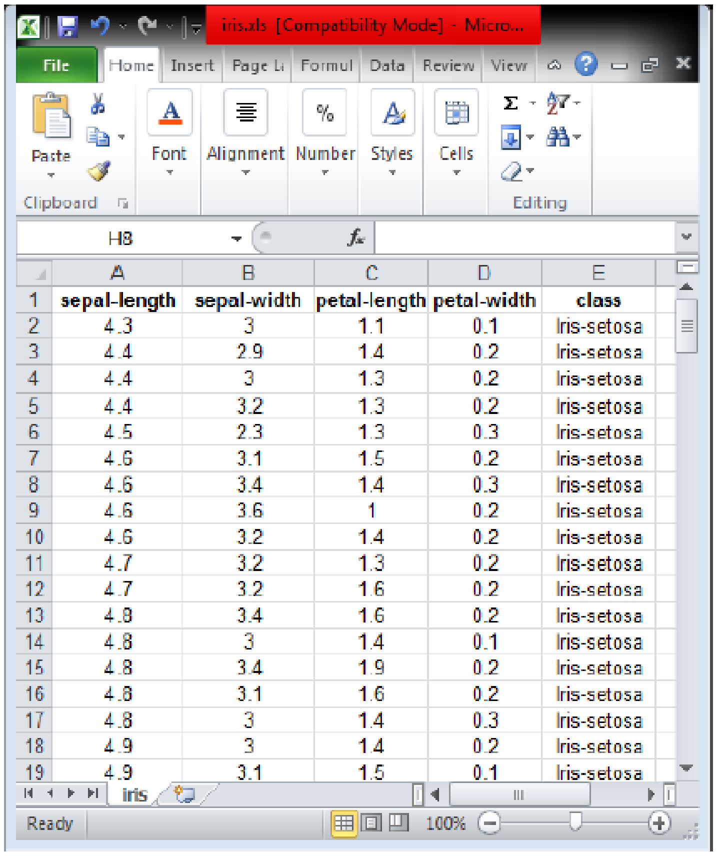
Data stored in an Excel worksheet can be easily converted into csv/ARFF format. The following steps are to be followed to convert a data spreadsheet into csv format:

1. Open the data spreadsheet in MS Excel.

2. Select File->Save as

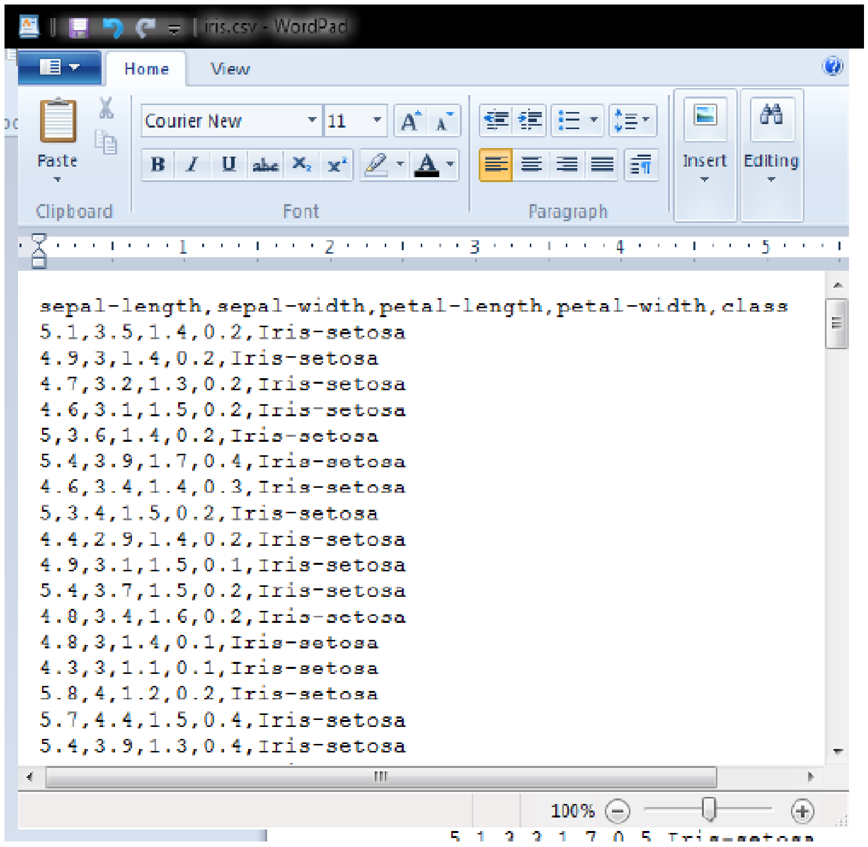
3. In Sava as type tab Select CSV (Comma Delimited)(*.csv)

4. The file has been saved in CSV format. You can load it in WEKA. The values in the first row will be taken as Attribute names and the subsequent rows will be treated as data instances.



	A	B	C	D	E
1	sepal-length	sepal-width	petal-length	petal-width	class
2	4.3	3	1.1	0.1	Iris-setosa
3	4.4	2.9	1.4	0.2	Iris-setosa
4	4.4	3	1.3	0.2	Iris-setosa
5	4.4	3.2	1.3	0.2	Iris-setosa
6	4.5	2.3	1.3	0.3	Iris-setosa
7	4.6	3.1	1.5	0.2	Iris-setosa
8	4.6	3.4	1.4	0.3	Iris-setosa
9	4.6	3.6	1	0.2	Iris-setosa
10	4.6	3.2	1.4	0.2	Iris-setosa
11	4.7	3.2	1.3	0.2	Iris-setosa
12	4.7	3.2	1.6	0.2	Iris-setosa
13	4.8	3.4	1.6	0.2	Iris-setosa
14	4.8	3	1.4	0.1	Iris-setosa
15	4.8	3.4	1.9	0.2	Iris-setosa
16	4.8	3.1	1.6	0.2	Iris-setosa
17	4.8	3	1.4	0.3	Iris-setosa
18	4.9	3	1.4	0.2	Iris-setosa
19	4.9	3.1	1.5	0.1	Iris-setosa

Figure 1. Iris, xis.



```
sepal-length,sepal-width,petal-length,petal-width,class
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3,1.4,0.1,Iris-setosa
4.3,3,1.1,0.1,Iris-setosa
5.8,4,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
```

Figure 2. Iris.csv.

2.2. Conversion from xls/xlsv to ARFF format

The following steps are to be followed to convert a data spreadsheet into ARFF format:

1. Repeat the above four steps to convert the data spreadsheet into csv format.

2. Open the above-created csv file in Notepad/Wordpad so that it can be edited.

3. At the top of the file use @relation tag to define the name of the dataset.

4. Delete the first row of the file which corresponds to attribute names.

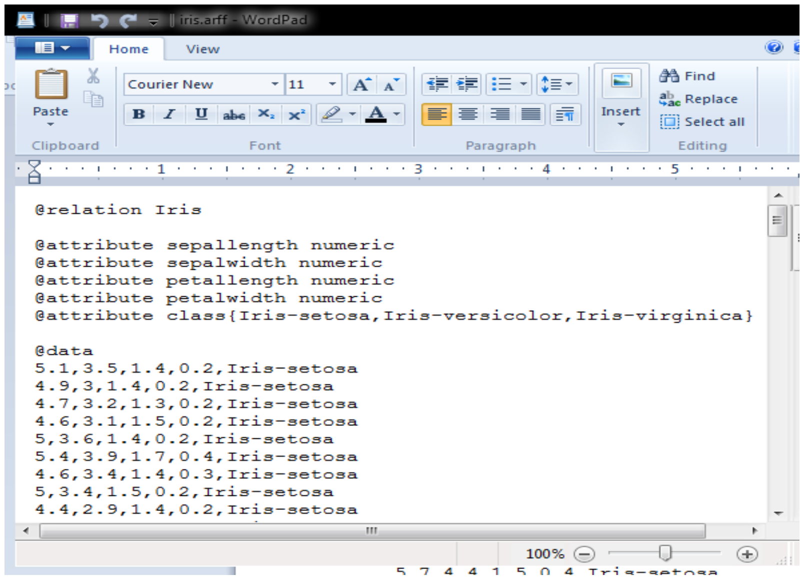
5. Use @attribute tag followed by attribute name and type as explained in the previous section to define all the attributes of the dataset.

6. Insert @data tag one row above the data instances.

7. Save and close the file.

8. Rename the file and change its extension from csv to ARFF.

9. The file is ready to be loaded into WEKA.



```
@relation Iris

@attribute sepallength numeric
@attribute sepalwidth numeric
@attribute petallength numeric
@attribute petalwidth numeric
@attribute class{Iris-setosa,Iris-versicolor,Iris-virginica}

@data
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
```

Figure 3. Iris.arff.

3. The WEKA GUI

When WEKA is launched, the Weka GUI chooser appears on the screen. The GUI Chooser includes of five buttons corresponding to the five major Weka applications and four menus. A screen shot of Weka GUI chooser is given in Figure 4. The buttons can be used to start the following applications:

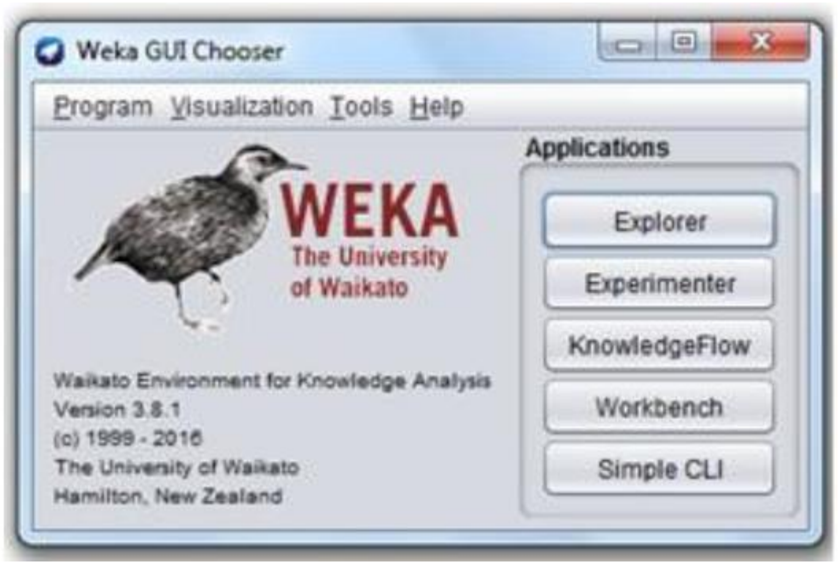


Figure 4. WEKA GUI.

1. Explorer: The main access to WEKA's GUI is obtained through its Explorer. The researcher can use menu selection and form filling to access all the facilities provided by the Explorer.

2. Experimenter: It provides a platform for experimentation. Analysis can be made by applying multiple algorithms on the same dataset to check which algorithm is performing better for a given dataset. An algorithm can be evaluated on different datasets. So, the experimenter allows large scale comparison of predictive performances of learning algorithms.

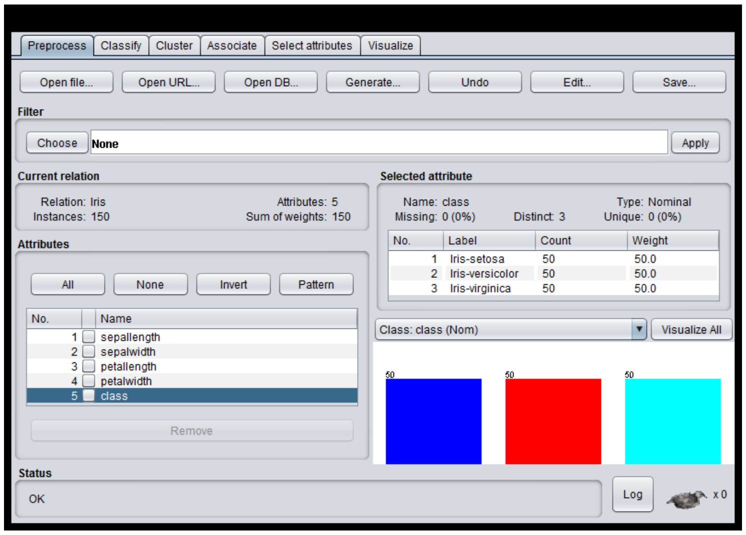
3. Knowledge Flow: An alternative option to the Explorer is the Knowledge Flow interface. Users can lay out and connect widgets representing WEKA components such as the data sources, data sinks, evaluation, visualization, etc. It allows us to layout filters, classifiers, evaluators interactively on a 2-D campus.

4. Workbench: It is a new feature that has been launched from Weka 3.8.0 (Bouckaert, et al., 2016). It is an all-in-one application which combines all the other Weka GUI's within the perspective of user-selection.

5. Simple CLI: It permits a direct execution of WEKA commands by providing a simple command-line interface.

3.1. Explorer

The researcher can use menu selection and form filling to access all the facilities provided by the Explorer. The basic functions supported are given by six tabs at the top of the Explorer. Initially, the user is in Preprocess tab.



Filter: None

Current relation

Relation: Iris Attributes: 5
Instances: 150 Sum of weights: 150

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute

Name: class Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count	Weight
1	Iris-setosa	50	50.0
2	Iris-versicolor	50	50.0
3	Iris-virginica	50	50.0

Class: class (Nom)

Status

OK

Figure 5. The Weka Explorer.

3.1.1. Preprocessor

Of all the tabs only the first tab is active at the start of the explorer, because dataset is required to be opened before the exploration of the data. At the click of the button “Open file”, a standard dialog appears from which a file can be selected, when the file is loaded, the screen shown in Figure appears which displays information of the dataset. The dataset has five attributes and one hundred and fifty instances (center left). The name of the attributes is also given. At the click of the attributes, the information about that attribute appears. In Figure, the class attribute is selected. In the selected attribute section, the entire information about class attribute is shown on the right side – the name of the attribute is class, data type is nominal, there are no missing values, it has three distinct values and none of the instances has a unique value. The count of different nominal values is also given. By default, Weka picks the last attribute as the class attribute, though a different class attribute can also be selected.

An attribute can be deleted at the click of its checkbox and the "Remove" button. "All" selects all the attributes, "None" selects none, the current selection can be inverted using "Invert", and "Pattern" selects the attributes that have similarity with a user-supplied regular expression. The "Undo" button helps in reverting the changes. The "Edit" button provides an editor which allows the inspection of the data, search for particular values and their editing, and in deleting the instances and the attributes. Corresponding context menus appears at the right click on values and column headers (Witten, Frank, & Hall, 2011).

Filters can be defined in preprocess section. This may transform the data in various ways. The required filters are set up using the filter box. At its left is a "Choose" button which helps in the selection of one of the filters in WEKA. After this selection, the name of filter and its options appear in the field next to the "Choose" button. A click on this box brings up a "Generic Object Editor" dialog box (Bouckaert, et al., 2016).

Filters are of two types – Supervised and Unsupervised, each of which are further of two types – Attribute and Instance. The Attribute filters can be applied to the attributes of the dataset such as to add an attribute, remove an attribute, etc. Unsupervised filters can be applied to the instances of the dataset such as to randomize the instances or to remove instances.

3.1.2. Classifier

"Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown"(Han, Kamber, & Pei, 2011). Classification is a data mining task of predicting the value of a categorical variable. This is done by building a model based on one or more numerical and/or categorical variables. The classification process uses a training set to train a model to predict unseen data. In this study we train, evaluate, and apply a classifier to classify flowers into their appropriate species. **Error! Reference source not found.** shows that data from file Iris-train.arff is loaded and a learning algorithm such as OneR, IBK (K Nearest Neighbours), J48 (C4.5), etc can be applied to this training set to create a model. This model can be applied to predict the class new data sets whose class attribute is unknown.

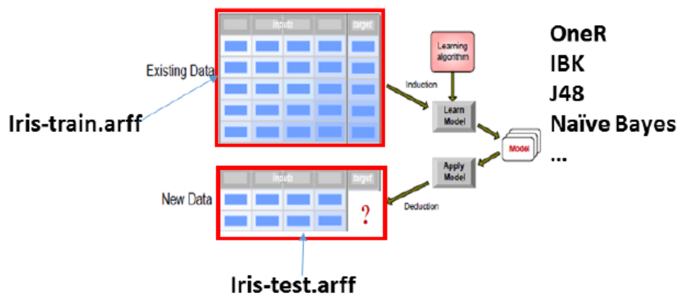


Figure 6. Building a Classifier.

The classify section contains the "Classifier" box at its top. The classifier box contains a text field which shows names of the classifier selected, and its options. The options can be configured by clicking the text box and then the dialog box. The result of applying the chosen classifier will be tested according to the options that are set by clicking in the "Test options" box. There are four test modes:

1. Use Training Set: In this case all the dataset is used to build a model for the classifier. Based on the classifier, the classes of all the instances in the dataset are evaluated again and the accuracy of the classifier is evaluated based on how well it predicts the class of instances against their actual classes.

2. Supplied Test Set: All the dataset is used to build a model for the classifier. A separate test file is loaded and the evaluation of the classifier is done on the basis of the authenticity of the prediction of the class of a set of instances loaded from that file. The file to test on can be chosen by clicking the set button.

3. Cross-validation: In 10-fold Cross – validation, the dataset is divided into ten sets. In the first run, nine sets are used for training the classifier and the tenth set is used for testing the classifier. In the next run, another set is held out as the test data and the remaining nine sets are used as training sets. Thus each set is held out in turn as the testing set and the process is repeated ten times. So each data set used once for testing and nine times for training. The results i.e. the accuracy and estimate of error are calculated as an average of these ten runs. Weka then runs the algorithm for the eleventh time to produce a classifier that can be used to predict classes. Stratified cross-validation ensures that each fold has the right proportion of each class value. Weka uses Stratified cross-validation by default.

4. Percentage Split: In Percentage split, the data set is divided into two parts based on value entered in the % field. By default, the value in the % field is 66. This means that 66% of the data is held out and the remaining data is used as the test set. The classifier is evaluated on how well it predicts the class of the test data.

The above four options are used only as a method of evaluation. The final model is always built using all the training data.

Testing options may be set at the click of the "More options..." button (Bouckaert, et al., 2016). Some of these options are – Output Model, Output per-class stats and Output confusion matrix.

Once the classifier, test options and class have all been set, the learning processis started at the click of the "Start" button. At the completion of the training, the "Classifier output area" is filled with text which describes the results of training and testing. It gives a new entry in the "Result list" box. At the click of the left mouse button and Alt and Shift in the text area, a dialog ox appears which saves the displayed output in different formats. The output is split into several sections:

(1) Run information: This gives information about the algorithm (the .class file executed), the relation name, the number of instances, the number and name of the attributes and the test mode used.

(2) Classifier model (full training set): This is a textual representation of the classification model which is produced using the full training data.

(3) The results of the chosen test mode are thus divided into the following:

(i) Summary: This lists the statistics which summarises the ability of the classifier to predict the true class of instances accurately under the chosen test mode.

(ii) Detailed Accuracy by Class: This displays a detailed per-class break down of the prediction accuracy of classifier.

(iii) Confusion Matrix: It tells about the number of instances assigned to each class. Its elements depict the number of test examples. The rows of the matrix show the actual class of the instances whereas the column shows the predicted class. The diagonal elements show the number of correctly classified instances.

(iv) Source code (optional): This section gives lists of the Java source in the "More options" dialog for "Output source code".

After the run is complete, a new entry appears in the "Result list" box. After training several classifiers, several entries are included in the result list. At the click on the entries, a user can flick back and forth between a number of generated results. The selected entry from the results can be deleted at the right click. Results can be viewed in a separate window and can be saved. Similarly, the model can be saved, loaded and reevaluated on a selected test set and perform a number of other tasks.

4. Using Filters to create your own Training set and Test set

It is always better to evaluate a classifier with a test set that is different from the training set. If we use the same training set as a test set also, the results can be misleading. So given there are enough instances, we can divide our dataset into a training set and a test set. Both the training set and test set should have a right proportion of each class value. We can use Weka filters to divide a given data set into training and test sets using the following steps:

1. Load the data set by pressing "Open" File button and choosing the appropriate file. In this case, we will load Iris.arff file. The file has 150 instances of data.

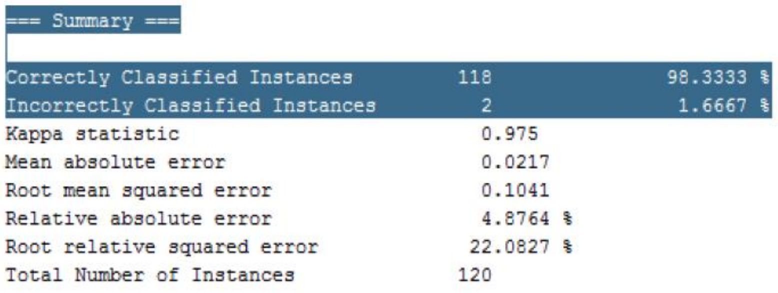
2. Choose Filters->Unsupervised->Instance->Randomize in the "Filters" tab in Preprocess panel. This filter shuffles the order of instances of the dataset that are passed through it. You can set the properties of the filter by clicking once on the selected filter. For example, in this case, you can choose a different seed for randomization. Click "Apply" to apply this filter on the data set. You can always revert the changes applied to a data set by pressing "Undo" button at the top of Preprocess panel. The changes applied to the data set can be seen in the "Edit" panel by pressing the "Edit" button which is there at top along the "Undo" button. Edit panel can also be used to apply changes to data items.

3. Choose Filters->Unsupervised->Instance->RemovePercentage in the filters tab in Preprocess panel. This filter removes a given percentage of a dataset. By default, the percentage is set to 50. In this case set the percentage to 20. Click "Apply" to apply this filter on the data set. You will see in the Current relation information that now there are 120 instances of data left in the data set. Save these instances as your training data set by pressing "Save" button and saving the file as Iris-train.arff.

4. Load the original Iris.arff data set again. Choose Filters->Unsupervised->Instance->RemovePercentage in the filters tab in Preprocess panel. Again set the percentage to 20 and set the "invertSelection" property to "true". Click "Apply" to apply this filter on the data set. You will see in the Current relation information that now there are 30 instances of data left in the data set. Save these instances as your test data set by pressing "Save" button and saving the file as Iris-test.arff.

5. Using Test data set to Evaluate a Classifier

1. Load Iris-train.arff in the Preprocess panel. Move to "Classify" panel, set the "Test options" to "Use training set" and build a classifer by pressing "Start" button. If you go to the summary in "Classifier output", you will see that when this classifier is evaluated on the training set, the accuracy is 98.33%.



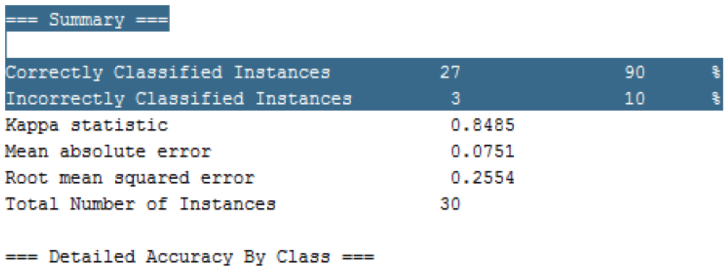
```
=== Summary ===

Correctly Classified Instances         118               98.3333 %
Incorrectly Classified Instances         2                1.6667 %
Kappa statistic                          0.975
Mean absolute error                      0.0217
Root mean squared error                  0.1041
Relative absolute error                  4.8764 %
Root relative squared error             22.0827 %
Total Number of Instances              120
```

Figure 7. Classifier Output when used on Training Set.

2. Now set the "Test options" to "Supplied test set" and load Iris-test.arff by clicking "Set…" button. Click on "More options" button in "Test options" and in "Output predictions" choose "Plain text". Right click on the classifier built in last step in Result list and select "Re-evaluate model on current test

set” from the menu. Now you will see that percentage accuracy is 90%. So we can clearly see that when the same training data set is used for evaluating a classifier, the results can be misleading. The evaluation of the classifier on a separate test set is always better. If you want to display the actual class and the predicted class of the test data, then click on “More options” button in “Test options” and in “Output predictions” choose “Plain text”. Right click on the classifier in “Result list” and select “Re-evaluate model on current test” set from the menu.

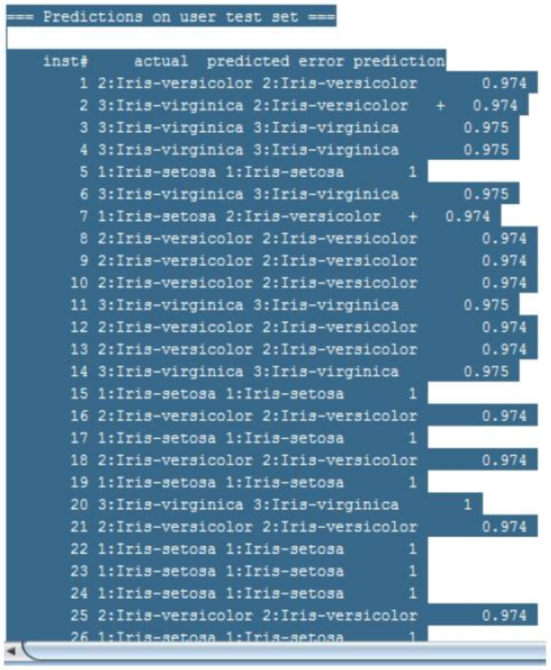


```
=== Summary ===

Correctly Classified Instances          27               90      %
Incorrectly Classified Instances         3               10      %
Kappa statistic                          0.8485
Mean absolute error                      0.0751
Root mean squared error                  0.2554
Total Number of Instances               30

=== Detailed Accuracy By Class ===
```

Figure 8. Classifier Output for Test Set.



```
=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1 2:Iris-versicolor 2:Iris-versicolor       0.974
        2 3:Iris-virginica 2:Iris-versicolor   +   0.974
        3 3:Iris-virginica 3:Iris-virginica       0.975
        4 3:Iris-virginica 3:Iris-virginica       0.975
        5 1:Iris-setosa 1:Iris-setosa        1
        6 3:Iris-virginica 3:Iris-virginica       0.975
        7 1:Iris-setosa 2:Iris-versicolor   +   0.974
        8 2:Iris-versicolor 2:Iris-versicolor       0.974
        9 2:Iris-versicolor 2:Iris-versicolor       0.974
       10 2:Iris-versicolor 2:Iris-versicolor       0.974
       11 3:Iris-virginica 3:Iris-virginica       0.975
       12 2:Iris-versicolor 2:Iris-versicolor       0.974
       13 2:Iris-versicolor 2:Iris-versicolor       0.974
       14 3:Iris-virginica 3:Iris-virginica       0.975
       15 1:Iris-setosa 1:Iris-setosa        1
       16 2:Iris-versicolor 2:Iris-versicolor       0.974
       17 1:Iris-setosa 1:Iris-setosa        1
       18 2:Iris-versicolor 2:Iris-versicolor       0.974
       19 1:Iris-setosa 1:Iris-setosa        1
       20 3:Iris-virginica 3:Iris-virginica       1
       21 2:Iris-versicolor 2:Iris-versicolor       0.974
       22 1:Iris-setosa 1:Iris-setosa        1
       23 1:Iris-setosa 1:Iris-setosa        1
       24 1:Iris-setosa 1:Iris-setosa        1
       25 2:Iris-versicolor 2:Iris-versicolor       0.974
       26 1:Iris-setosa 1:Iris-setosa        1
```

Figure 9. Predicted class of Tuples in Test Set.

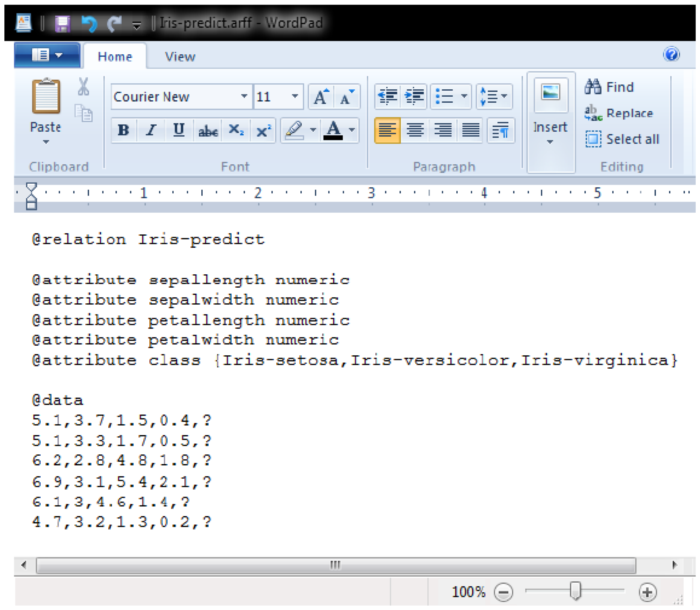
6. Using Classifier to make Predictions on New Data

Once we have built a model, it can be used to classify new unseen data:

1. Create a file that contains new data i.e. the data for which class is unknown. This file should have the structure similar to the file which is used for learning the model. The only difference is that here we don't know the value of class variable. Replace the value of the class attribute with "?" for all instances, question mark represents a missing value in Weka.

2. At the right click of menu on the "Result list" panel, the saved model is loaded.

3. In the "Test Options", select "Supplied test set"and load Iris-predict.arff by clicking "Set…" button. Click on "More options" button in "Test options" and in "Output predictions" choose "Plain text". Right click on the classifier loaded in step 2 in Result list and select "Re-evaluate model on current test set" from the menu.

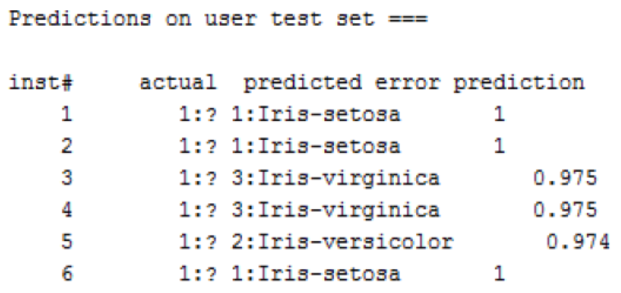


```
@relation Iris-predict

@attribute sepallength numeric
@attribute sepalwidth numeric
@attribute petallength numeric
@attribute petalwidth numeric
@attribute class {Iris-setosa,Iris-versicolor,Iris-virginica}

@data
5.1,3.7,1.5,0.4,?
5.1,3.3,1.7,0.5,?
6.2,2.8,4.8,1.8,?
6.9,3.1,5.4,2.1,?
6.1,3,4.6,1.4,?
4.7,3.2,1.3,0.2,?
```

Figure 10. Iris-predict.arff.



```
Predictions on user test set ===

inst#     actual  predicted error prediction
    1        1:? 1:Iris-setosa       1
    2        1:? 1:Iris-setosa       1
    3        1:? 3:Iris-virginica       0.975
    4        1:? 3:Iris-virginica       0.975
    5        1:? 2:Iris-versicolor       0.974
    6        1:? 1:Iris-setosa       1
```

Figure 11. Predicted class of Unknown tuples.

Here **Error! Reference source not found.**shows the file Iris-predict.arff. The class values for these instances are unknown. So they are marked as "?" (Missing values). **Error! Reference source not found.** shows the result of "Predictions on user test set" in "Classifier output" panel.

7. Conclusion

The paper explores the data mining tool – WEKA. The different file formats that can be used with WEKA tool are discussed. The default file format of WEKA is ARFF. The paper shows how to store data in ARFF format and how to convert data from other file formats into ARFF format. The paper carries an in depth study of WEKA GUI. Classification task consists of examining the features of a newly presented object and assigning a predefined class to it. The study shows how to create and evaluate a classifier using WEKA tool. The study has been carried out using IRIS dataset which is provided along the WEKA tool. When the model is checked using the same dataset that is used to build the model, the accuracy is found to be 98.33%. This reduces to 90% when different dataset is used to check the accuracy of the model. Therefore, the dataset is divided into the training dataset and the test dataset; the first dataset is brought in use to create the model, and the accuracy of the model is checked with the second dataset. The study also shows how the model can be used for the prediction of the class of those tuples/objects, the class label of which is not known.

References

[1] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, et al. WEKA Manual for Version 3-8-1. Hamilton, New Zealand: University of Waikato. (2016).

[2] N. L. Costa, L. A. Llobodanin, I. A. Castro and R. Barbosa, Using Support Vector Machines and Neural Networks to Classify Merlot Wines from South America, Information Processing in Agriculture (2019), 265-278.

[3] W. M. Dlamini, A Data Mining Approach to Predictive Vegetation Mapping Using Probabilistic Graphical Models, Ecological Informatics 6 (2011), 111-124.

[4] M. H. Dunham, Data Mining Introductory and Advanced Topics, Pearson (2012).

[5] H. A. Edelstein, Introduction to Data Mining and Knowledge Discovery. Potomac, MD, USA: Two Crows Corporation, (2005).

[6] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, From Data Mining To Knowledge Discovery in Databases. AI Magazine 17(3) (1996), 37-54.

[7] R. Ghorbani and R. Ghousi, Predictive Data Mining Approaches in Medical Diagnosis: A Review of Some Diseases Prediction. International Journal of Data and Network Science (2019), 47-70.

[8] A. Haghverdi, B. Ghahraman, B. G. Leib, I. Pulido-Calvo, M. Kafi, K. Davary, et al. Deriving Data Mining and Regression Based Water-salinity Production, Computers and Electronics in Agriculture (2014), 68-75.

[9] M. F. Hall, The WEKA Data Mining Software: An Update, ACM SIGKDD explorations newsletter 11(1) (2009, June), 10-18.

[10] J. Han, M. Kamber and J. Pei, Data Mining Concepts and Techniques, Waltham, MA, USA: Morgan Kaufmann Publishers. (2011).

[11] F. Hoppner, Association Rules, In L. Roach and O. Maimon, Data Mining and Knowledge Discovery Handbook (pp. 299-320). Springer.

[12] D. T. Larose, Discovering Knowledge in Data: An Introduction to Data Mining. Hoboken, New Jersey: John Wiley & Sons. (2014).

[13] O. Maimon and L. Rokach, Weka-A Machine Learning Workbench for Data Mining. In E. Frank, M. Hall, G. Holmes, R. Kirkby, B. Pfahringe, I. H. Witte, et al., Data Mining and Knowledge Discovery Handbook (pp. 1269-1277). London: Springer. (2010).

[14] T. M. Mitchell, Machine Learning. Chennai: McGraw Hill Education. (2018).

[15] A. Mucherino, P. J. Papaorgji and P. M. Pardalos, Data Mining in Agriculture. Springer. (2009).

[16] D. L. Olson and L. G. Descriptive Data Mining, In Descriptive Data Mining (pp. 129-130). Singapore: Springer. (2019).

[17] L. Rokach, A Survey of Clustering Algorithms, In L. Rokach and O. Maimon, Data Mining and Knowledge Discovery Handbook (pp. 269-298). Springer. (2010).

[18] L. Rokach and O. Maimon, Classification Trees, In L. Rokach and O. Maimon, Data Mining and Knowledge Discovery Handbook (pp. 148-174). Springer. (2010).

[19] L. Rokach and O. Maimon, Supervised Learning, In L. Rokach and O. Maimon, Data Mining and Knowledge Discovery Handbook (pp. 133-147). Springer. (2010).

[20] L. J. Rutkowski, Stream Data Mining: Algorithms and their Probabilistic Properties. Springer Nature. (2019).

[21] P. Sebastiani, M. M. Abad and M. F. Ramoni, Bayesian Networks, In O. Maimon and L. Rokach, Data Mining and Knowledge Discovery Handbook (pp. 175-208). Springer. (2010).

[22] G. M. Weiss and B. D. Davison, Data Mining, In H. Bidgoli, Handbook of Technology Management (pp. 542-555). Hoboken, NJ: John Wiley and Sons. (2010).

[23] I. H. Witten, E. Frank and M. A. Hall, Data Mining Practical Machine Learning Tools and Techniques. Burlington, USA: Elsevier. (2011).