COMPUTING SCIENCE

Data mining and machine learning in e-Science Central using Weka

Dominic Searson

TECHNICAL REPORT SERIES

No. CS-TR-1454 February 2015

TECHNICAL REPORT SERIES

No. CS-TR-1454 February, 2015

Data mining and machine learning in e-Science Central using Weka

D. Searson

Abstract

Weka is a mature and widely used set of Java software tools for machine learning, data-driven modelling and data mining – and is regarded as a current gold standard for the practical application of these techniques.

This paper describes the integration and use of elements of the Weka open source machine learning toolkit within the cloud based data analytics e-Science Central Platform. The purpose of this is to extend the data mining capabilities of the e-Science Central platform using trusted, widely used software components in such a way that the non-machine learning specialist can apply these techniques to their own data easily. To these ends, around 25 Weka blocks have been added to the e-Science Central workflow palette. These blocks encapsulate (1) a representative sample of supervised learning algorithms in Weka (2) utility blocks for the manipulation and pre-processing of data and (3) blocks that generate detailed model performance reports in PDF format. The blocks in the latter group were created to extend existing Weka functionality and allow the user to generate a single document that allows model details and performance to be referenced outside of e-Science Central and Weka.

Two real world examples are used to demonstrate Weka functionality in e-Science Central workflows: a regression modelling problem where the objective is to develop a model to predict a quality variable from an industrial distillation tower, and a classification problem, where the objective to is predict cancer diagnostics (tumours classified as 'Malignant' or 'Benign') based on measurements taken from lab cell nuclei imaging. Step by step methods are used to show how these data sets may be modelled, and the models evaluated, using blocks in e-Science Central workflows.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

SEARSON, D.
Data mining and machine learning in e-Science Central using Weka
[By] D. Searson
Newcastle upon Tyne: Newcastle University: Computing Science, 2015.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1454)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1454

Abstract

Weka is a mature and widely used set of Java software tools for machine learning, data-driven modelling and data mining – and is regarded as a current gold standard for the practical application of these techniques.

This paper describes the integration and use of elements of the Weka open source machine learning toolkit within the cloud based data analytics e-Science Central Platform. The purpose of this is to extend the data mining capabilities of the e-Science Central platform using trusted, widely used software components in such a way that the non-machine learning specialist can apply these techniques to their own data easily. To these ends, around 25 Weka blocks have been added to the e-Science Central workflow palette. These blocks encapsulate (1) a representative sample of supervised learning algorithms in Weka (2) utility blocks for the manipulation and pre-processing of data and (3) blocks that generate detailed model performance reports in PDF format. The blocks in the latter group were created to extend existing Weka functionality and allow the user to generate a single document that allows model details and performance to be referenced outside of e-Science Central and Weka.

Two real world examples are used to demonstrate Weka functionality in e-Science Central workflows: a regression modelling problem where the objective is to develop a model to predict a quality variable from an industrial distillation tower, and a classification problem, where the objective to is predict cancer diagnostics (tumours classified as 'Malignant' or 'Benign') based on measurements taken from lab cell nuclei imaging. Step by step methods are used to show how these data sets may be modelled, and the models evaluated, using blocks in e-Science Central workflows.

About the authors

Dominic Searson holds an M.Eng. degree in Chemical Engineering and a PHD in machine learning and multivariate analysis from Newcastle University. He is the author of the popular GPTIPS open source software platform for data mining and non-linear predictive modelling and has research interests in machine learning, evolutionary computation, data-driven modelling, complex systems and multivariate statistical modelling.

Suggested keywords

REGRESSION
CLASSIFICATION
CLOUD COMPUTING

Data mining and machine learning in e-Science Central using Weka

Dominic Searson

School of Computing Science

Newcastle University

2015

dominic.searson@ncl.ac.uk

Abstract

Weka is a mature and widely used set of Java software tools for machine learning, data-driven modelling and data mining – and is regarded as a current gold standard for the practical application of these techniques.

This paper describes the integration and use of elements of the Weka open source machine learning toolkit within the cloud based data analytics e-Science Central Platform. The purpose of this is to extend the data mining capabilities of the e-Science Central platform using trusted, widely used software components in such a way that the non-machine learning specialist can apply these techniques to their own data easily. To these ends, around 25 Weka blocks have been added to the e-Sc workflow palette. These blocks encapsulate (1) a representative sample of supervised learning algorithms in Weka (2) utility blocks for the manipulation and pre-processing of data and (3) blocks that generate detailed model performance reports in PDF format. The blocks in the latter group were created to extend existing Weka functionality and allow the user to generate a single document that allows model details and performance to be referenced outside of e-Sc and Weka.

Two real world examples are used to demonstrate Weka functionality in e-Science Central workflows: a *regression* modelling problem where the objective is to develop a model to predict a quality variable from an industrial distillation tower, and a *classification* problem, where the objective to is predict cancer diagnostics (tumours classified as 'Malignant' or 'Benign') based on measurements taken from lab cell nuclei imaging. Step by step methods are used to show how these data sets may be modelled, and the models evaluated, using blocks in e-Science Central workflows.

1 Introduction

The objective of this paper is to demonstrate the use of the Weka data mining toolkit within the e-Science Central (e-Sc) software platform and to show the use of these tools on some simple but illustrative data sets. The intention is that readers will then be able to use Weka blocks within e-Sc on their own data mining tasks with minimal difficulty.

However, the actual data mining/machine learning algorithms used are not discussed in any substantial detail as this is not intended to be a primer in this area. A good starting point for an introduction to data mining – providing a balance between theoretical and practical considerations - is the book 'Data mining: practical machine learning tools and techniques' by Witten and Franke [1].

This document begins with a brief description of the e-Science Central software platform, blocks and workflows in Section 2. This is followed by a short introduction to Weka and some 'Weka specific' data and file formats in Section 3. Section 4 contains a reasonably detailed discussion of the different e-Sc block categories in Weka - namely: 'Tools', 'Regression' and 'Classification' and their functionality.

It is explained how these blocks may be chained together to solve tasks related to data pre-processing and data mining. Finally Sections 5 and 6 contain examples using 'real' data showing how full data-mining workflows may be constructed – and the results interpreted - for a regression problem and a binary classification problem.

2 e-Science Central

e-Science Central is a scalable data storage and data analytics platform written by researchers at the School of Computing Science, Newcastle University. It is an open source software project that has been under active, continuous development since 2008 and has been used successfully on a variety of academic and commercial data-intensive scientific research projects e.g. [2].

The e-Science Central platform allows users to store and analyse data securely – either privately or by sharing it with colleagues within projects. The e-Science Central platform is server/browser based making it useable anywhere and on almost any operating system.

The core of e-Science Central is the workflow engine – this facilitates the analysis and processing of data either locally or on cloud computing resources (e.g. Amazon WS and Microsoft Azure) – making it highly scalable. Workflows are sequential data processing pipelines, which are drawn graphically by users in a GUI, and can be stored and shared in e-Science Central like any other piece of data.

e-Science Central is free to download from the BitBucket repository at https://bitbucket.org/digitalinstitute/esciencecentral .

2.1 Workflows and blocks

The basic unit of functionality in an e-Science Central workflow is the 'block'. Each block performs a distinct data processing function (e.g. load data from text file, plot data etc.) Blocks are selected from block palettes and chained together to form a workflow.

A typical workflow consists first of blocks that load data into a workflow, followed sequentially by blocks that perform task-specific data processing and finally by blocks that write the results of the workflow to data storage.

Blocks can have both input and output ports and are joined together by means of connecting lines (connectors) – representing the transfer of data from one block to another via input and output ports. When a workflow is run its blocks are executed sequentially so that a block only outputs the result of its processing when it is complete. Then the next sequential block in the workflow is executed. When the workflow is complete the user may inspect the results using a browser.

2.2 e-Science Central workflow data types

Data is transferred from one block to another via ports and connectors in three main formats. The type(s) of data that each block accepts and/or exports via connectors is block specific and these data types are not interchangeable. For instance, a block that expects one data type (e.g. `FileWrapper`) via an input port cannot accept another data type via that port.

The three principal workflow data types are:

1) A `FileWrapper` – this data type encapsulates one or more files. These are typically text files (e.g. CSV files) containing experimental data or report files (e.g. PDF files or graphs) generated within the workflow.

2) An `ObjectWrapper` – this data type encapsulates a serialised (binary) Java class and is used to pass classes (e.g. a class representing a model) from one block to another. For instance, in the Weka block set it is used to pass 'trained' model objects from one block to another.

3) A `DataWrapper` – this data type is used to transfer 'columns' of data (e.g. numerical or textual) data between blocks. Each `DataWrapper` contains one or more columns – which can be accessed by name or position. This is a particularly useful construct because scientific data is often represented as matrices and vectors and there is a natural correspondence between the matrix/vector and `DataWrapper` representations. The e-Science Central platform contains numerous blocks for the manipulation of data within and between `DataWrappers`, e.g. `ColumnSelect`, `ColumnJoin`, `DataShuffle` etc.

3 WEKA

3.1 Introduction

Weka [3] (Waikato Environment for Knowledge Analysis) is a suite of open source software tools for data mining, machine learning, data analysis and predictive modelling tasks. It is written in Java and provides facilities and algorithms for data loading, data pre-processing, statistics, classification, regression, clustering, and visualization [1].

Weka is currently at version 3.6 and has been in development at the Machine Learning Group at the University of Waikato, New Zealand since 1994. It is free under the Gnu General Public License (GPL). Weka is extremely extensive and contains a diverse set of tools and algorithm - for example, there are currently more than 75 regression and classification algorithm implementations.

Weka can be used in two basic modes:

1) *Using the standalone Weka 'Knowledge Explorer' GUI*. This is the 'basic' mode of usage. The GUI facilitates data loading, visualisation, pre-processing and modelling tasks.
2) *Integrating Weka Java classes and code using the Java API*. This is the advanced mode of usage, allowing the integration of Weka components in other software frameworks and platforms. This permits far more powerful configuration options than the GUI mode – but a degree of machine learning and Java development expertise is required to integrate the Weka components correctly. This is the method by which Weka components have been integrated into the e-Science Central platform.

3.2 Weka terminology and data types.

In Weka terminology:

Variables (whether they are inputs or outputs) are referred to as *attributes*.

Attributes can be of type NUMERIC (integers or real valued variables, e.g. 3.4, 29.999, -10), or of type NOMINAL (e.g. category labelled data such as 'Blue', 'Yellow', 'Malignant', 'Benign' etc.). Other Weka types are DATE and STRING (any other unspecified non-numeric type).

Note that data types must normally be either NUMERIC or NOMINAL for usage with Weka machine learning/modelling classes.

A set of corresponding observations – which in CSV format would normally be a row of comma separated data - is called an *instance*.

3.3 Data formats

In terms of manipulating and processing data, Weka operates almost exclusively on a flat data file format called ARFF (Attribute-Relation File Format), which acts as the common currency of Weka.

The ARFF file is similar to the common CSV file format but has the advantages that it is easy to read for humans and computers, that missing values and sparse valued data are supported in a natural consistent way and that metadata is stored in a simple manner. ARFF is an increasingly popular data exchange format. For instance, a number of recent datasets added to the UCI Machine Learning Repository are in ARFF format as well as CSV.

The majority of Weka classes (whether they be simple data processing steps or the implementations of complex machine learning algorithms) operate directly on an ARFF file as an input and produce an ARFF file as output - often adding annotations to the output ARFF `@relation` property indicating the data processing used.

An example of an ARFF file is shown in **Fig. 1** below.

The `@relation` property defines a name for the data. Each `@attribute` property defines the name of a variable and its Weka data type.

The `@data` property defines the start of the data. Each subsequent line until the end of the file contains a data instance with the data ordered the same way as the attribute definitions (similar to a row of data in CSV files).

```
% A comment line

@RELATION 'some data'      Text describing the data.
@ATTRIBUTE x1  NUMERIC              Defines a variable named 'x1' as numeric.
@ATTRIBUTE x2  NUMERIC     Defines a variable named 'x2' as numeric.
@ATTRIBUTE y   {class1, class2}  Defines the 'values' a nominal variable 'y' can take.

@DATA
-10,12.4343,32.7463,class2A data instance (row)
22,-0.9821,12012.4299,class1
872,1.98,5.3762,class1
```

Fig. 1. Example of the ARFF data format. Text in italics is explanatory and not part of the file format.

Note that in ARFF files the modelled/output variable is not explicitly specified but in Weka the *last attribute* defined in the ARFF file is by default the modelled variable (in the above example it is *y* – a nominal/category valued variable that can take on the values 'class1' and 'class2' only). Similarly, in e-Science Central Weka blocks it is implicitly assumed that the last attribute defined is the one that is to be modelled.

To facilitate working with ARFF files, workflow blocks have been created to convert CSV data sources to ARFF format (see Section 4.2).

4 Weka in e-Science Central

The block palette in e-Science Central is divided into sub-categories where blocks providing related functionality are grouped. The Weka block palette is shown below in **Fig. 2**. Clicking on the required block and dragging it into the workflow editor allows that block to be added to a workflow.

4.1 Weka block functionality

The Weka palette comprises around 25 blocks – representing a significant expansion of the existing e-Science Central block palette. These are further categorised into the following sub-categories: 'Tools', 'Regression' and 'Classification'. Weka Tool blocks are discussed in 4.2. Regression is discussed in Section 4.3 and classification in Section 4.4.

Most blocks have user parameters that can be set, and Weka modelling blocks have machine learning algorithm specific parameters that can be adjusted by the user in order to optimise algorithm performance on the data being mined. For instance consider the `NeuralNet` Weka modelling block in **Fig. 3**. This contains seetings such as Epochs (the number of iterations over the training data) and Hidden Neurons (the number of non-linear processing nodes in the hidden layer of the feed-forward artifical neural net).

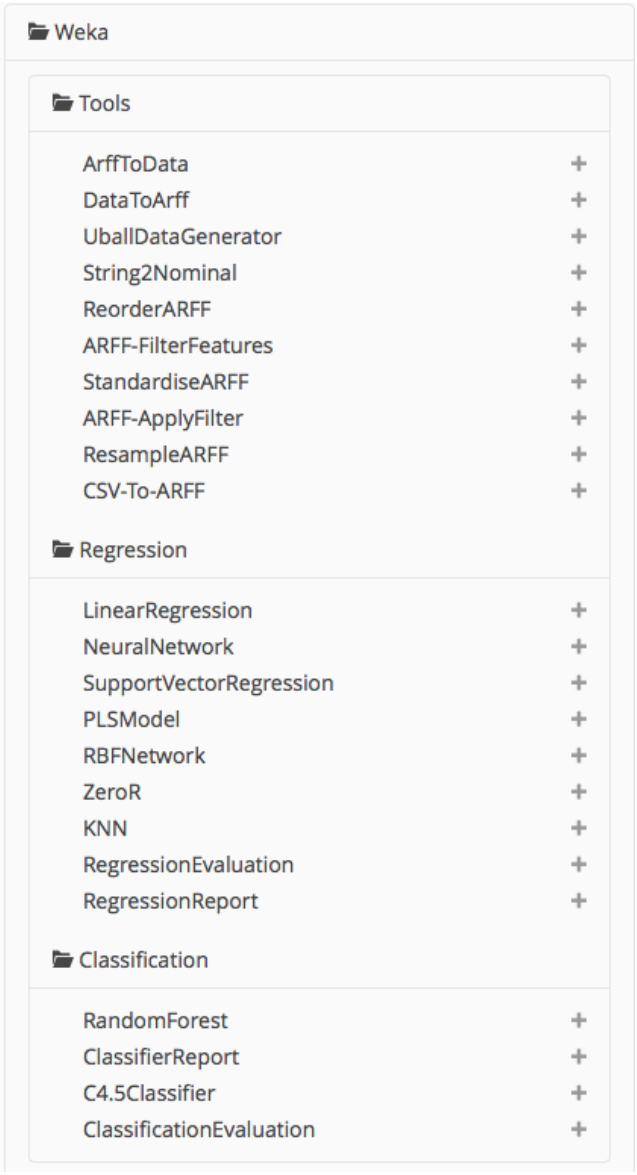


Fig. 2. The Weka block palette in the e-Science Central workflow editor.

Each Weka modelling block has different parameters which relate to the implementations of the underlying machine learning algorithms, a discussion of which is beyond the scope of this article, but they are however detailed in pop-up help when mouse hovering on the block user fields. Existing online Weka documentation also provides a more detailed description for each of the methods/parameters for a block.

4.2 Tools

The Weka Tools palette includes blocks for the loading and manipulation of data in ARFF format. E.g. `CSV-To-ARFF`. This block loads data in a CSV file format and converts it into ARFF format. The output port of this block is a `FileWrapper` connection containing an ARFF file.

Similarly, the `DataToARFF` block has one input port that accepts a connection containing a `DataWrapper` and it attempts to convert the data in it into ARFF format exported in a `FileWrapper`. The `ARFFToData` block does the opposite conversion.

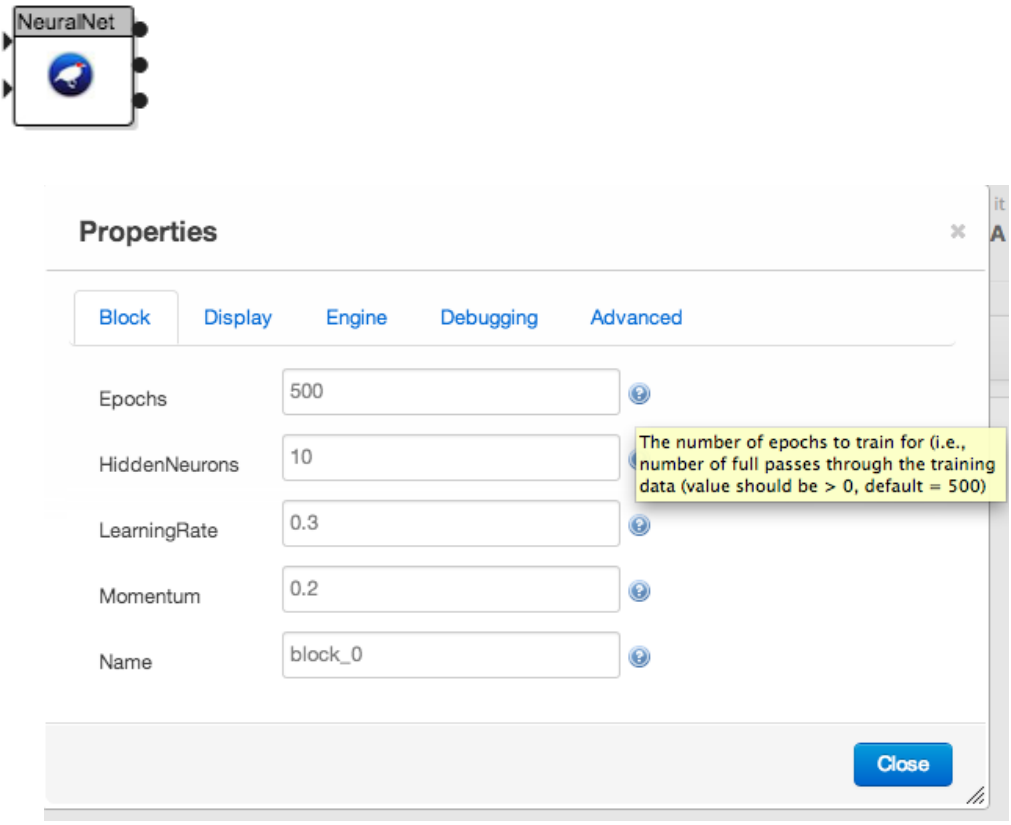


Fig. 3. User settable parameters of the Weka `NeuralNet` regression modelling block.

In addition, there are blocks for the simple pre-processing of data. For example, `ResampleARFF` – which accepts a `FileWrapper` containing an ARFF file through an input port - resamples the data (with replacement) and exports the sampled data as an ARFF file through a `FileWrapper` output port. Likewise, the `ReorderARFF` block rearranges the order of attributes in an ARFF file and the `String2Nominal` block converts any attributes of Weka data type STRING in an ARFF file to data type NOMINAL.

An example of a small workflow that loads a CSV file, converts it to ARFF format, resamples it and exports the resampled data as an ARFF file is shown in **Fig. 4**.[1]

[1] Resampling a training data set is often used to build more robust models by combining models built using different subsets of the training data. This is known as 'bagging' (bootstrap aggregating).

Fig. 4. Use of Weka blocks to convert a CSV data file to ARFF format, resample it and export the resampled data as another ARFF file. Here, each connector between blocks transfers a `FileWrapper` containing an ARFF file.

Finally, within the Weka Tools palette, there are blocks for the filtering of data for tasks such as feature selection and numerical transformation of the data. These blocks also export (as an `ObjectWrapper`) the filter so that it can be applied to other data (e.g. testing data) using the `ARFF-ApplyFilter` block.

Current filter blocks are `ARFF-FilterFeatures` (this performs supervised feature/attribute selection) and `StandardiseARFF` (this converts numerical attributes to zero mean and unit variance). E.g. for feature selection the `ARFF-FilterFeatures` block is typically applied to the training data and then the `ARFF-ApplyFilter` block is used to apply the exported filter in an `ObjectWrapper` for use on the testing data). An example of this is shown below in **Fig. 5**.

The `ARFF-FilterFeatures` block uses Weka machine learning algorithms to automatically select the most relevant *x* attributes to include as inputs to a Weka data mining/modelling block. This may be required for certain data sets with a large number of input attributes because many modelling methods perform poorly when there are a large number of input variables (only some of which are usually significantly correlated to the *y* variable that we wish to predict.)

Feature selection is often regarded as an integral part of the model building process and hence is performed on the training data to learn which features to keep; this is then applied to the testing data using the trained filter.

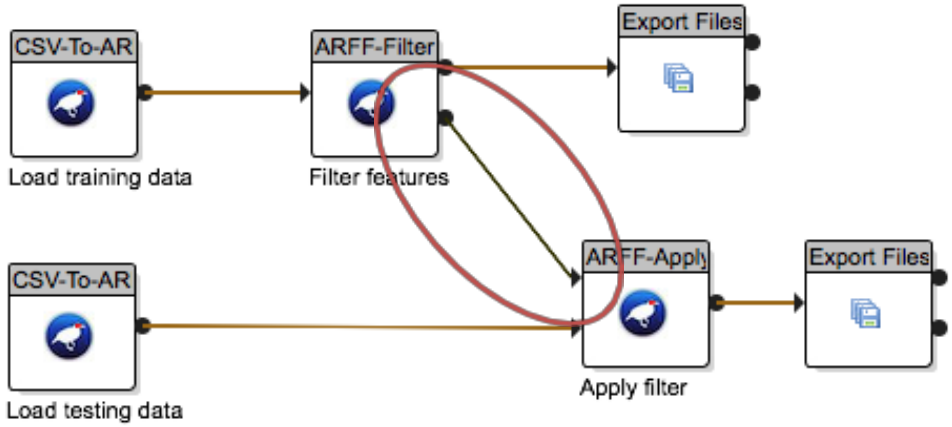


Fig. 5. Example of using the Weka `ARFF-ApplyFilter` block to apply a filter created by `ARFF-FilterFeatures` on training data to testing data.

4.3 Regression

The majority of blocks in the regression palette are *predictive modelling* (supervised learning) blocks that are used to learn to predict a numeric output variable y using n numeric input variables $x_1, \cdots, x_n$ on a set of m training data observations. Each y is assumed to be an unknown function f of the corresponding x plus some error term.

Hence, using $\mathbf{y}$ and $\mathbf{X}$ to denote matrix/vector form[2]:

$\mathbf{y} = f(\mathbf{X}) + \mathbf{e}$ where $\mathbf{e}$ is a vector of errors (e.g. noise or other unmodelled effects).

We want to find some function $f'(\mathbf{X})$ that closely approximates $f(\mathbf{X})$ across the $\mathbf{X}$, $\mathbf{y}$ data that we have collected. This is the training part of data mining. Hence, we pick a regression model structure $f'(\mathbf{X})$ (e.g. linear regression, neural network) and find the parameters of the model structure that minimises $\mathbf{e}$ in some sense, typically the sum of squared errors (SSE) $\mathbf{e}^T\mathbf{e}$.

Some regression blocks assume a linear dependence on the x inputs (e.g. `LinearRegression` and `PLSModel`) and others allow a non-linear dependence to be assumed (e.g. `NeuralNet` and `SVMRegression`). Each regression modelling block always has the same 2 input ports and 3 output ports – allowing different regression blocks to be dropped in and out of a workflow with minimal inconvenience.

Regression block Input ports

The first (top left) block input port is always a `DataWrapper` connection containing the m values $\mathbf{y}$ of the y variable to be modelled. The second (bottom left) is always a `DataWrapper` connection containing the corresponding m values $\mathbf{X}$ of the n input variables.

For example, the Weka Tools `ARFFToData` block can be used to convert an imported ARFF file to the `DataWrapper` format, and then the `ColumnSelect` block can be used to select the y data and the x data as `DataWrapper` input connections to the Weka regression modelling block. This is illustrated below in **Fig. 6** with a Weka `LinearRegression` block.

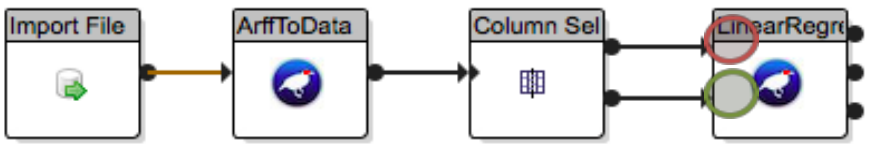


Fig. 6. Example of using the Weka `ARFFToData` block and `ColumnSelect` block to supply training data (y – red circle and x – green circle) to the input ports of a Weka `LinearRegression` modelling block.

[2] Here the bold $\mathbf{y}$ is used to mean a (m x 1) column vector containing m observed values of the variable y and $\mathbf{X}$ is used to denote an (m x n) matrix containing the m observed values of $x_1, \cdots, x_n$

Regression block output ports
Each regression modelling block has 3 output ports:

The first output port (top right) is a `DataWrapper` containing two columns: the *m* observations of the modelled variable **y** and the *m* corresponding predictions of this variable $\hat{\mathbf{y}}$. Internally, each modelling block performs an optimisation that minimises the errors **e** between the actual and predicted values. In the case of the `LinearRegression` block, the sum of squared errors is minimised. This `DataWrapper` is used to pass **y** and $\hat{\mathbf{y}}$ to a `RegressionReport` block.

The second output port (middle right) is a `FileWrapper` containing a report text file generated by the underlying Weka Java regression class for the block. The contents of the file are block specific. In the case of `LinearRegression`, for instance, the file contains the coefficients of the trained linear model. The contents of the report file are for information only and are not required by any other blocks.

The final output port (bottom right) is an `ObjectWrapper` containing the trained model object. The principal utility of this is to export the trained model so that it can be evaluated on data that was not used to build the models (i.e. a testing data set).

Good performance on training data does not necessarily indicate a good model (overfitting is common) and so evaluation on unseen testing data is crucial. This is accomplished by connecting the trained model object to a `RegressionEvaluation` block, which accepts an `ObjectWrapper` connector containing a trained model as indicated below in **Fig. 7**. It also accepts connections for `DataWrappers` containing the modelled variable *y* and the inputs *x* for the testing data in much the same way as the regression modelling blocks do for training data.

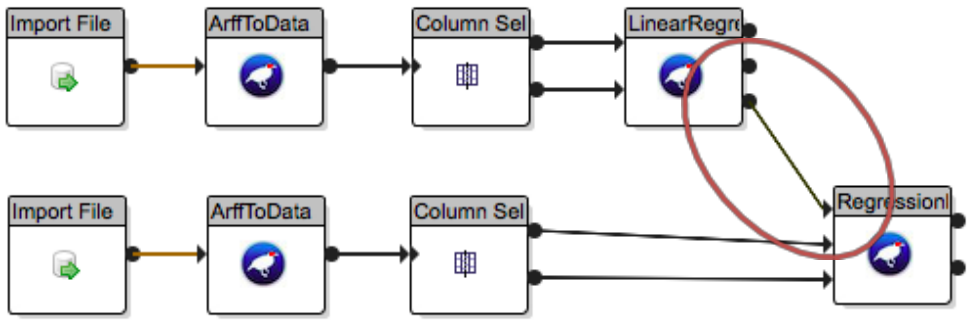


Fig. 7. Example of exporting (highlighted in red) a trained `LinearRegression` model object in an `ObjectWrapper` connector and applying it to new testing data using the `RegressionEvaluation` block.

Regression model reports
As discussed, each regression modelling block (and the `RegressionEvaluation` block) outputs a connector (from the top right of the block) containing a `DataWrapper` with two columns. The first column in the `DataWrapper` contains the observed values **y** and the second contains the predicted values $\hat{\mathbf{y}}$. This `DataWrapper` can be connected to a `RegressionReport` block to generate a PDF report containing

key model details and model performance metrics.[3] This will be discussed further by means of an example with real data in Section 5.

4.4 Classification

The blocks in the Weka classification palette are also *predictive modelling* (supervised learning) blocks. The main difference between these and the regression blocks is that – instead of predicting a numerical output *y* using numerical inputs *x* – we are trying to predicted a nominal (class labelled or categorical) output y using either numerical or nominal *x* variables (or a mixture of both). E.g. in regression modelling, *y* takes on a value in a continuous numerical range whereas in classification *y* takes only categorical values like 'high', ' medium' or 'low'. The categories that *y* can take on are problem dependent.

However – aside from this difference – the way that the blocks are connected together (and the ordering of input/output connectors) is identical to that of blocks from the regression palette. Current classification modelling blocks are `C4.5Classifier` which induces a decision tree on the training data using Quinlan's C4.5 learning algorithm [5] and `RandomForest` which induces an ensemble of decision trees based on samples of the training data [6].

Analogous to the regression palette, trained classification models are evaluated on testing data using the `ClassificationEvaluation` block and PDF performance reports may be generated using the `ClassificationReport` block.

5 Regression problem: distillation tower

This is data from a real distillation tower from the Dow chemical company. The purpose of a distillation tower is to separate a liquid feedstock containing multiple chemical components into 2 or more fractions. Each fraction in the feedstock has a different volatility (boiling point) and a tower contains a number of internal stages where separation occurs. The more easily boiled fraction is extracted from the top of the tower and the heavier less volatile fraction(s) at the bottom of the tower.

Distillation towers are widely used in the chemical and process industries and are energy intensive and expensive to run. Hence – it is imperative that they are run efficiently. In practice, they are tightly controlled using industrial temperature, flow and pressure control systems. To facilitate the control of distillation towers it is extremely useful to have a numerical predictive model of how the behaviour of the tower changes according to changes in inputs (temperatures, flow rates, pressures). It is not usually practical to derive a first principles mathematical model of the dynamic behaviour of a tower; so often purely empirical (data-driven) models must be created to run a tower safely and efficiently.

Next - it is shown how a predictive data-driven model of data acquired from a real distillation tower can be generated using Weka regression blocks. It is also shown how the predictive performance of models can be evaluated using Weka blocks.

5.1 Data description

In the distillation tower data set, there are 57 input (*x*) variables and one output (*y*) variable. The data is pre-partitioned into training and testing data sets. The training

[3] In fact, a `DataWrapper` containing $\mathbf{y}$ and $\hat{\mathbf{y}}$ from any source (not just Weka modelling blocks) can be connected to a `RegressionReport` block. However, when Weka blocks are used additional detail is added to the PDF report.

data (used to build a model) is contained within a CSV file called `dowTrain.csv` (containing 747 rows of data) and the testing data (used to evaluate a model) is contained within `dowTest.csv` (containing 319 rows of data).

Dow has removed the measurement units associated with the input *x* variables for reasons of commercial sensitivity, but the output variable *y* is a % concentration of propylene (a volatile organic compound which is derived from oil and natural gas processing and is an important precursor in the production of thermosetting plastics).

Each CSV file contains a header row containing the variable names ($x_1, \cdots, x_{57}, y$) and each following row contains a measurement of each of the *x* variables and the corresponding *y* variable.

5.2 Objective

Predict the propylene concentration (*y*) at the top of the tower using 57 rapidly sampled *x* variables (flows, pressures etc.) from tower instrumentation (i.e. create a 'soft-sensor' or 'inferential estimator' of *y*). The modelled *y* output is numeric (as are all of the *x* inputs) so this may be regarded as a regression problem. In this case no data pre-processing is performed and the data is loaded directly from CSV files without using any intermediate transformations requiring Weka Tools blocks. In this case the use of a simple linear regression model is illustrated but the procedure is the same for other forms of regression block, e.g. neural net.

5.3 Procedure

The basic procedure for building a workflow for modelling the tower data follows. The final workflow is shown in **Fig. 8**.

Step 1. Load CSV training and testing data using separate `CSVImport` blocks.

Step 2. Use `ColumnSelect` blocks to extract the *y* modelled variable and the *x* input variables from the `DataWrapper` connectors exported from each `CSVImport` block.

Step 3. Connect the *y* and *x* output ports from the `ColumnSelect` blocks to a `LinearRegression` block (for the training data) and a `RegressionEvaluation` block (for the testing data).

Step 4. Connect the output port containing the trained model object from the `LinearRegression` block to the `RegressionEvaluation` block.

Step 5. Connect the `DataWrapper` output ports from `LinearRegression` and `RegressionEvaluation` (each containing $\mathbf{y}$ and $\hat{\mathbf{y}}$) to `RegressionReport` blocks.

Step 6. Connect the `FileWrapper` output ports from each `RegressionReport` block to an `ExportFiles` block. This allows the generated PDF model performance reports to be written to the workspace when the workflow has been run).

Step 7. Save and run the workflow.

Step 8. Evaluate the model performance on the training and testing data sets using the generated PDF reports.

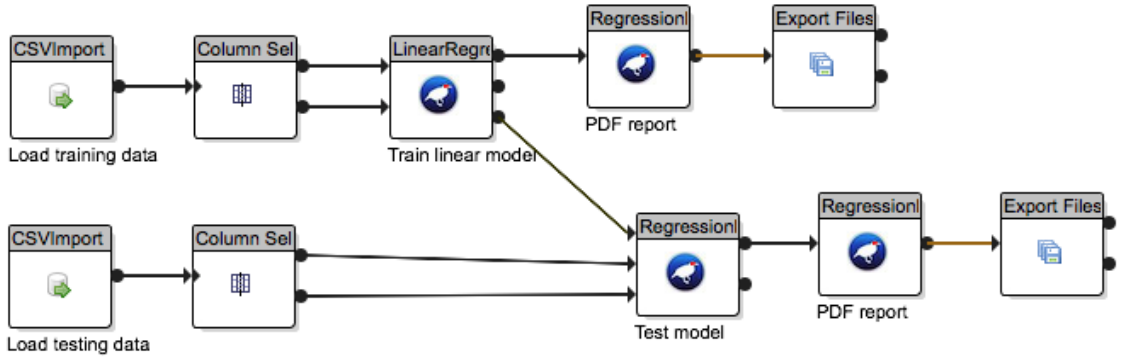


Fig. 8. Loading, modelling and generating linear regression model reports for the distillation tower data using Weka blocks in an e-Science Central workflow.

5.4 Analysis of results

The principal outputs of the workflow shown in **Fig. 9** are two PDF reports (training and testing data).

A regression performance report has two main sections. The first section is textual and summarises key model data and performance metrics. The second part comprises graphical data (such as a scatter plot of the actual and predicted values).

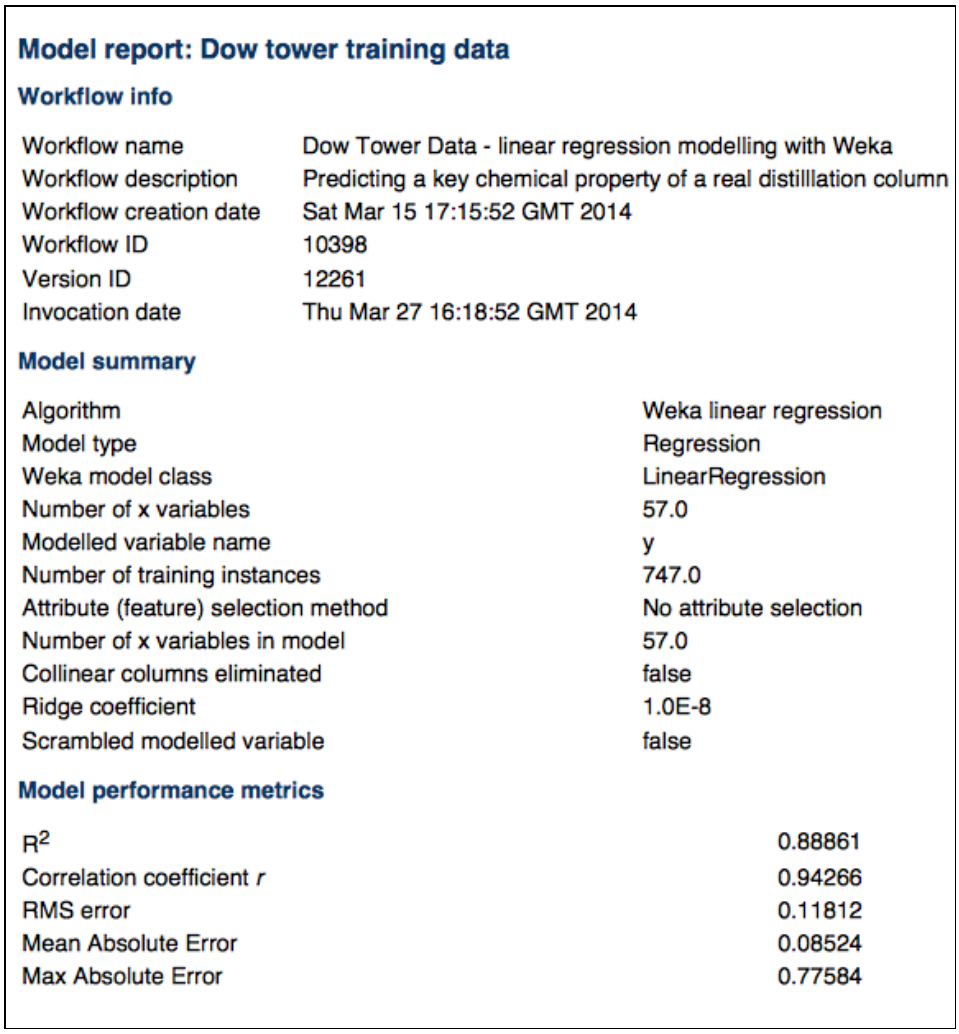
The first part of a typical PDF regression report for the distillation training data is shown below in **Fig. 9**. Graphical output is shown in **Fig. 10** (Scatter plot of *y* vs. predicted *y*), **Fig. 11** (Trendline plot of *y* and predicted *y*) and a Regression Error Characteristic (REC; see [4]) plot in **Fig. 12**.

The textual content of the PDF reports is as follows:

Workflow Info
It can be seen that the report section labelled 'Workflow Info' relates to the workflow used to model that data. This includes the name of the workflow and when it was run.

Model Summary
The content of this section relates basic information about the configuration of the model type used (e.g. the name of the underlying Weka Java class), the number of input *x* variables supplied to the model, the number of x variables actually used by the model and any other relevant parameters.

The exact content of this section depends on which model type was used. For instance, in **Fig. 9** a number of linear model configuration parameters are shown, e.g. 'Feature selection method' (which in this case refers to in block feature selection and is not used).



Model report: Dow tower training data

Workflow info

Workflow name	Dow Tower Data - linear regression modelling with Weka
Workflow description	Predicting a key chemical property of a real distilllation column
Workflow creation date	Sat Mar 15 17:15:52 GMT 2014
Workflow ID	10398
Version ID	12261
Invocation date	Thu Mar 27 16:18:52 GMT 2014

Model summary

Algorithm	Weka linear regression
Model type	Regression
Weka model class	LinearRegression
Number of x variables	57.0
Modelled variable name	y
Number of training instances	747.0
Attribute (feature) selection method	No attribute selection
Number of x variables in model	57.0
Collinear columns eliminated	false
Ridge coefficient	1.0E-8
Scrambled modelled variable	false

Model performance metrics

R^2	0.88861
Correlation coefficient *r*	0.94266
RMS error	0.11812
Mean Absolute Error	0.08524
Max Absolute Error	0.77584

Fig. 9. Model details and performance metrics from the `RegressionReport` block PDF on the distillation tower training data. Here an R^2 of 0.89 indicates good performance on the training data. A similar value on the testing data (0.87) indicates good model generalisation to unseen data.

Model performance metrics

This section of the report is the always the same regardless of which regression modelling method was used. The performance indicators used are:

R^2 – this is the coefficient of determination[4] – and represents the fraction of the variance of the data that was explained by the model.

Hence, $R^2 = 1$ would represent a perfect fit to the training data and $R^2 = 0.07$ would represent a very poor fit. Typical good models tend to vary from $R^2 = 0.75$ to $R^2 = 0.99$ (although this is greatly problem dependent.) R^2 is a commonly quoted performance metric because it is invariant with respect to the measurement units of the predicted *y* variable and the number of data points in a set.

In **Fig. 9**, only the results on the training data are shown (R^2 training = 0.88) but an R^2 of 0.87 was achieved on the testing data indicating a model that has explained much of

[4] R^2 is not – in general – the square of the correlation coefficient *r* and may take negative values for pathologically poorly fitting models.

the variation of the data and has generalised well to the testing data. It is necessary to ensure that similar performance on the testing data is achieved; otherwise it is likely that the model has over fitted the training data

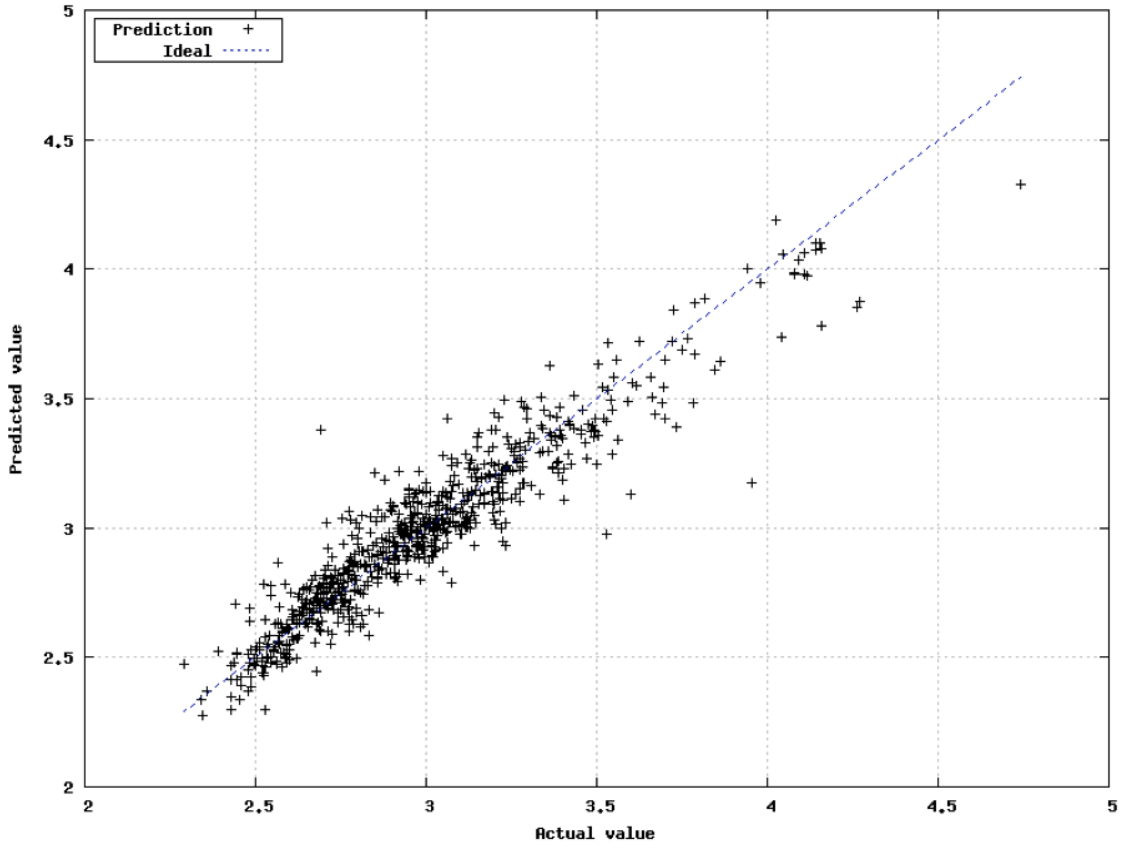


Fig 10. Scatter plot of *y* vs. predicted *y* from the `RegressionReport` block PDF on the distillation tower training data. Good predictions lie close to the blue dashed identity line ('ideal' predictions from a model with $R^2 = 1$ would all lie on the identity line).

Correlation coefficient *r* – this can vary between -1 and +1 and is the Pearson product moment correlation (between $\mathbf{y}$ and $\hat{\mathbf{y}}$) over the training data set. A high positive correlation is desirable.

RMS error – this is the root mean squared error over the data set. Small values are better. The RMS error is expressed in the units of the modelled variable *y*.

Mean absolute error – this is the mean of abs($\mathbf{e}$) and is expressed in the units of *y*.

Max absolute error – this is the maximum value of abs($\mathbf{e}$), i.e. the largest prediction error that occurred in the modelling of the data set.

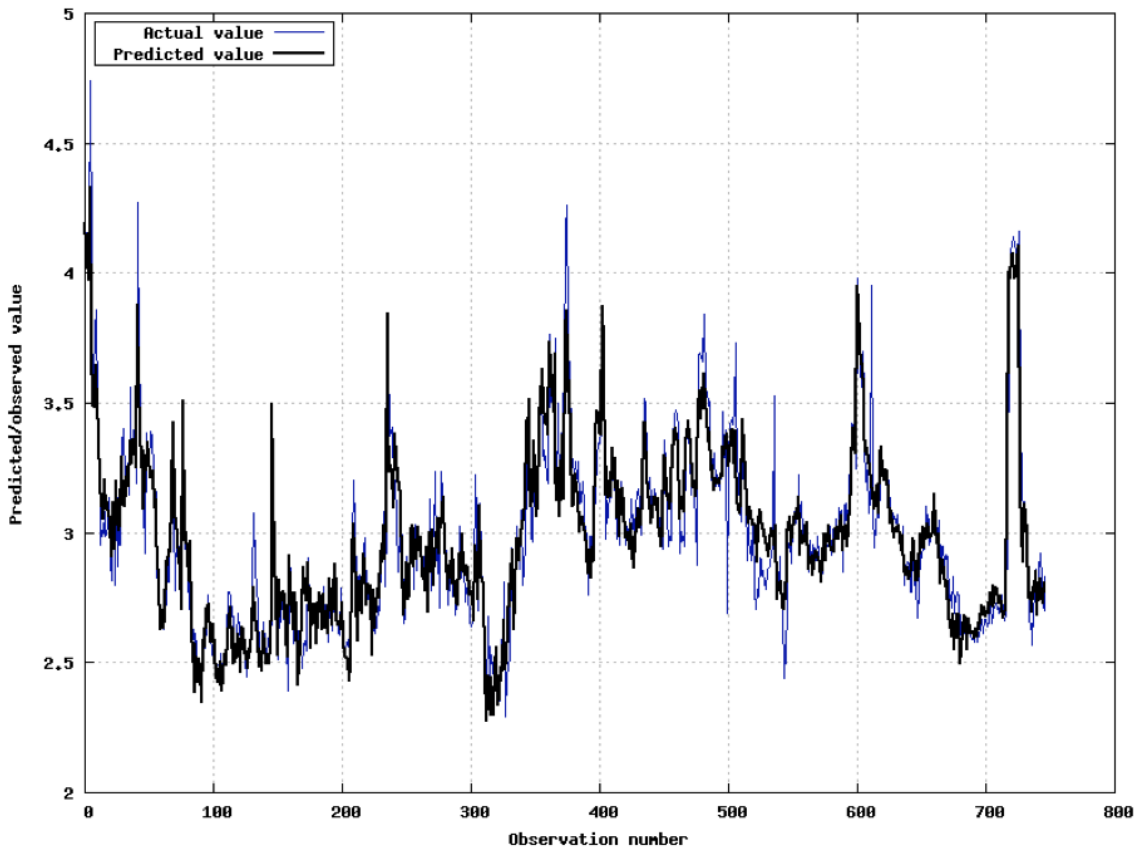


Fig. 11. Trend plots of *y* and predicted *y* from the `RegressionReport` block PDF report on the distillation tower training data.

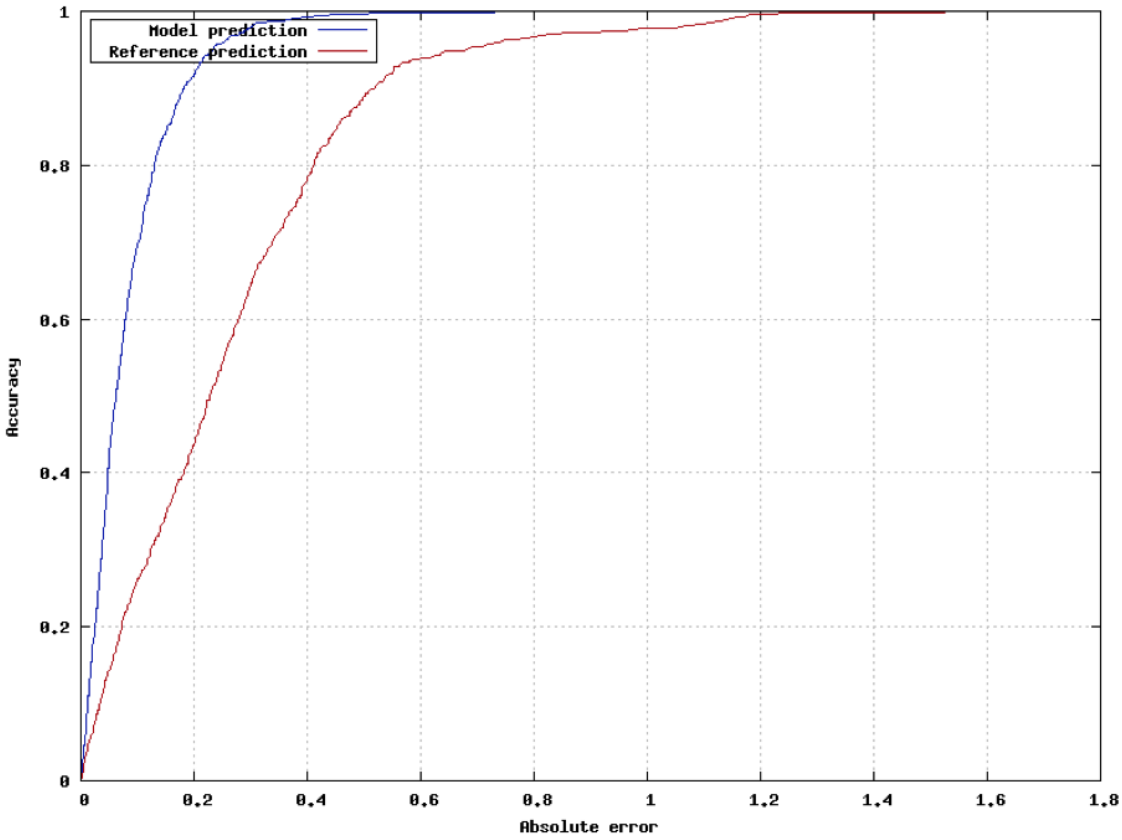


Fig. 12. A Regression Error Characteristic (REC) plot from the `RegressionReport` block PDF on the distillation tower training data. This shows (on the y-axis) the fraction (between 0 and 1) of data points predicted for a given absolute error value (on the x-axis). A naïve reference model (e.g. from the `ZeroR` regression block) is shown in red and the trained linear model in blue. A model is better than another if its REC curve lies above and to the left of the comparison model.

6 Classification problem: Wisconsin Breast Cancer Diagnostic (WBCD) data

This is real (anonymised) medical data from [7] obtained from the UCI Machine Learning repository at http://archive.ics.uci.edu/ml/. Each data record (instance) corresponds to a patient and contains a modelled y variable: a human expert diagnosis ('M' – malignant tumour or 'B' – benign) and 30 input variables derived from a fine needle aspirate (FNA) of a breast mass. The *x* variables are numeric characteristics of the cell nuclei present in the digitised image.

6.1 Objective

Predict the diagnosis ('M' or 'B') using the numeric nuclei image features x for unseen test data. The modelled *y* output is categorical/nominal so this may be regarded as a binary classification problem. In this case data is loaded directly from a single ARFF file and then randomly split into a training set and a testing set using e-Science Central and Weka blocks. Here, the C4.5 algorithm is used to create a decision tree to classify the data.

6.2 Data description

Part of the ARFF file containing the data is shown below in **Fig. 13**. The data contains 569 rows - 357 benign and 212 malignant diagnoses. Each row of data contains the diagnoses ('M' or 'B') followed by 30 comma separated numeric values of the corresponding image features x_1 to x_{30}.

```
@relation wbcd

@attribute y {M,B}
@attribute x1 numeric
@attribute x2 numeric
@attribute x3 numeric
@attribute x4 numeric
@attribute x5 numeric
@attribute x6 numeric
@attribute x7 numeric
@attribute x8 numeric
@attribute x9 numeric
@attribute x10 numeric
@attribute x11 numeric
@attribute x12 numeric
@attribute x13 numeric
@attribute x14 numeric
@attribute x15 numeric
@attribute x16 numeric
@attribute x17 numeric
@attribute x18 numeric
@attribute x19 numeric
@attribute x20 numeric
@attribute x21 numeric
@attribute x22 numeric
@attribute x23 numeric
@attribute x24 numeric
@attribute x25 numeric
@attribute x26 numeric
@attribute x27 numeric
@attribute x28 numeric
@attribute x29 numeric
@attribute x30 numeric

@data
M,
17.99,10.38,122.8,1001,0.1184,0.2776,0.3001,0.1471,0.2419,0.07871,1.095,0.9053,8.
589,153.4,0.006399,0.04904,0.05373,0.01587,0.03003,0.006193,25.38,17.33,184.6,201
9,0.1622,0.6656,0.7119,0.2654,0.4601,0.1189
```

Fig. 13. Part of the ARFF data file for the WBCD data set. Note that the modelled *y* variable (diagnosis) is the first declared attribute so a `ReorderARFF` block will be necessary to pre-process the data so that the *y* variable is last.

6.3 Procedure

The basic procedure for building a workflow for modelling the WBCD data using a `C4.5Classifier` block is as follows. The final workflow is shown in **Fig. 14**.

Step 1. Import the ARFF file using an `ImportFile` block.

Step 2. Reorder the ARFF file such that the *y* variable is the last declared attribute using a `ReorderARFF` block.

Step 4. Convert the ARFF file into a `DataWrapper` using an `ARFFToData` block.

Step 5. Use the `Subsample` block to sample approx. ¼ of the rows of data for testing data, keeping ¾ for training data.

Step 6. Use `ColumnSelect` blocks to extract the *y* and *x* data and to supply these to a `C4.5Classifier` (for training data) and a `ClassificationEvaluation` block (testing data).

Step 7. Connect the `C4.5Classifier` and `ClassificationEvaluation` to `ClassifierReport` blocks in order to generate PDF model performance reports.

Step 8. Save and run the workflow.

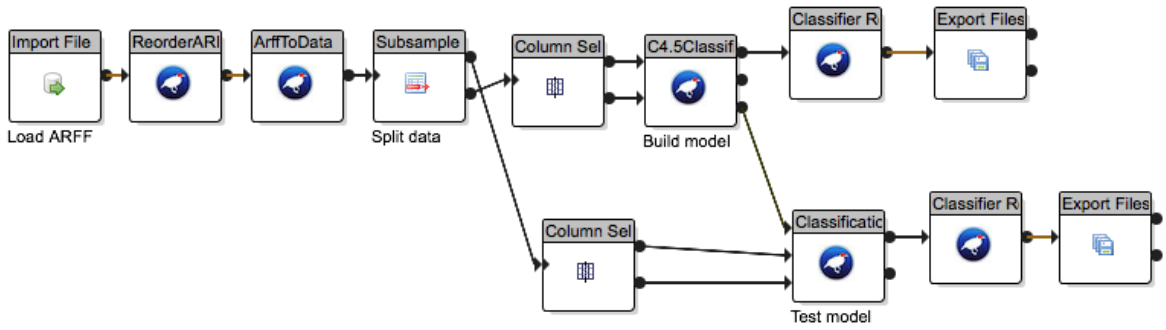


Fig 14. Loading, modelling and generating C4.5 classifier model reports for the WBCD data using Weka blocks in an e-Science Central workflow.

6.4 Analysis of results

The principal outputs of the workflow shown in **Fig. 14** are, again, two PDF reports (training and testing data).

A classification performance report is (currently) shorter than a regression report and only contains a textual section. A PDF of the results on the training data is shown in **Fig. 15**.

The 'Workflow info' and 'Classifier summary' sections are essentially directly equivalent to their counterparts from regression modelling, i.e. they contain basic configuration information, some descriptors of the model structure (in this case a decision tree) and some user parameters pertaining to the underlying Weka class.

Classifier performance metrics

This section in the PDF report details the performance of the classifier on the data set in question. This is quantified in terms of the number of data instances correctly classified as well as some other metrics for each *y* class/category. These are:

Recall – A number between 0 and 1 showing the fraction of all actual category C instances that were correctly predicted as being in category C. In **Fig. 14** the recall for both 'M' and 'B' diagnoses is close to 1, indicating very good performance.

Classifier report: Wisconsin BCD training data

Workflow info

Workflow name	Wisconsin Cancer Diagnostic - C4.5 modelling with Weka
Workflow description	Predicting breast cancer malignancy
Workflow creation date	Tue Mar 11 15:17:13 GMT 2014
Workflow ID	8711
Version ID	13450
Invocation date	Thu Apr 24 15:12:15 BST 2014

Classifier summary

Algorithm	C4.5 Decision Tree
Model type	Classification
Weka model class	J48
Unpruned trees only	false
Reduced error pruning	false
Binary splits only	false
Folds for reduced error pruning	5.0
Confidence interval for pruning	0.25
Number of x variables	30.0
Modelled variable name	y
Number of training instances	427.0
Scrambled modelled variable	false
Tree size	21.0
Number of leaves	11.0
Number of rules	11.0

Classifier performance metrics

Training instances correct	424
Training instances incorrect	3
Recall (class 'M')	0.98773
Precision (class 'M')	0.99383
F-measure (class 'M')	0.99077
Recall (class 'B')	0.99621
Precision (class 'B')	0.99245
F-measure (class 'B')	0.99433

Confusion Matrix

```
=== Confusion Matrix ===

   a   b   <-- classified as
 161   2 |   a = M
   1 263 |   b = B
```

Fig 15. Model details and performance metrics from the `ClassifierReport` block PDF on the WBCD training data. Good performance on the training data was achieved with only 3 instances (of 427) misdiagnosed. Class M = 'Malignant diagnosis'. Class B = 'Benign diagnosis'. Similar performance was achieved on the testing data.

Precision - A number between 0 and 1 showing the fraction of predicted Category C instances that were in fact in Category C. In **Fig. 15** this number is close to 1 for both categories, indicating good performance on the training data.

F-measure – this is a number between 0 and 1 and can be regarded as a weighted harmonic mean of the precision and recall scores.

Confusion matrix – this shows which categories the classifier has got 'confused'. The rows are the categories C (in this case 'M' and 'B') and the columns show how many of each category were in fact assigned by the classifier to the categories 'M' and 'B'. A good classifier should mainly contain entries on the leading diagonal of this matrix. Off-diagonal entries of the confusion matrix represent misclassifications.

7 Conclusions

It has been shown that gold standard machine learning software components for supervised machine learning have been incorporated within the cloud based e-Science central data analytics platform. This includes workflow blocks to: manipulate Weka ARFF files, split data into training and test sets, perform feature selection, train linear and non-linear predictive models on regression and classification tasks and generate detailed standalone model performance reports. This provides a useful and scalable data mining and modelling environment for the e-Science Central platform.

References

[1] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I.H., The WEKA data mining software: an update, SIGKDD Explorations, Volume 11, Issue 1, 2009.

[2] Watson P, Leahy D, Cala J, Sykora V, Hiden H, Woodman S, Taylor M, Searson D., *Cloud Computing for Chemical Activity Prediction.* Newcastle upon Tyne: School of Computing Science, University of Newcastle upon Tyne, 2011. School of Computing Science Technical Report Series **1242**.

[3] Witten, I.H, & Frank, E., Data mining: practical machine learning tools and techniques, 2nd Ed., ISBN-13: 978-0-12-088407-0, Elsevier, 2005.

[4] Bi, J. & Bennett, K.P., Regression error characteristic curves, Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

[5] Quinlan, Q., C4.5: Programs for machine learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[6] Breiman, L., Random forests, Machine Learning, Vol. 45, Issue 1, pp. 5-32, 2001.

[7] Street, W.N., Wolberg, W.H. & Mangasarian, O.L., Nuclear feature extraction for breast tumor diagnosis, IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, pp. 861-870, San Jose, CA, 1993.

Acknowledgements

I would like to thank my colleagues on the e-Science Central team at Newcastle University for technical assistance and advice.