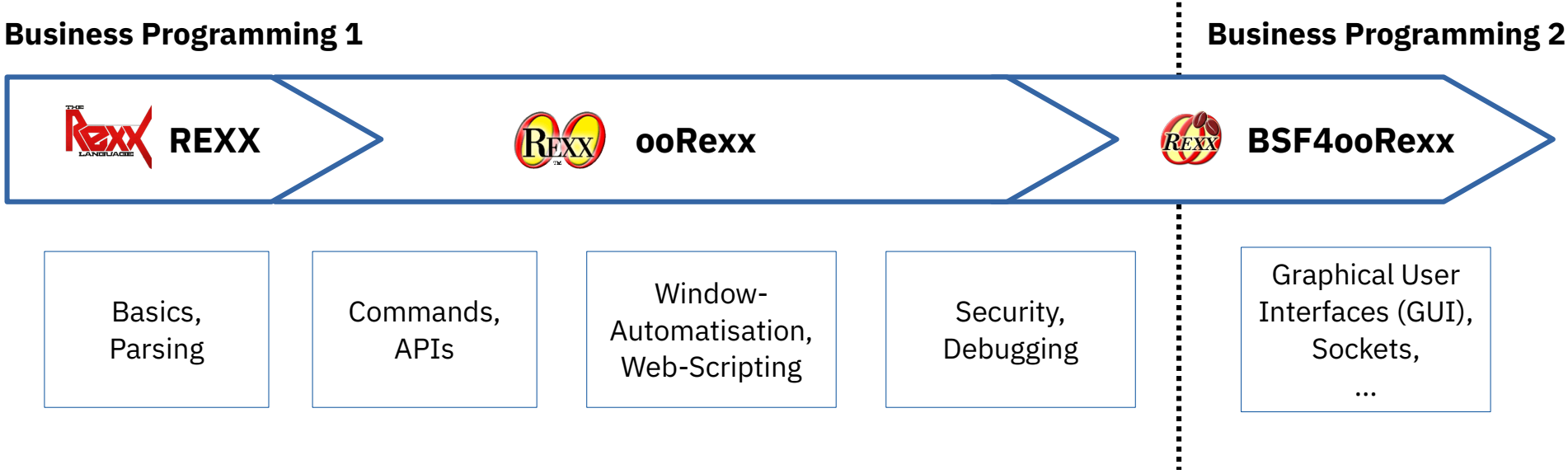
Artificial Intelligence - Machine Learning (ML) with ooRexx



Overview

- Introduction to machine learning (ML)
- Introduction to Weka (a powerful, complete Java infrastructure for ML)
- The Weka environment
- Weka resources (links to Weka, packages, tutorials, ...)
- Nutshell examples for employing Weka for ML
- Roundup

Prof. Rony G. Flatscher

Machine Learning (ML), 1
Early History of Machine Learning (1940s-1990s)

- Origins
  - 1940s–50s: Foundations in statistics, neuroscience, and computer science
  - Alan Turing (1950): “Can machines think?”
  - 1957: Frank Rosenblatt introduces the Perceptron (early neural network)
- Early Progress & Challenges
  - 1960s–70s: Rule-based AI (artificial intelligence) and symbolic systems dominate
  - 1980s: Revival of neural networks with backpropagation
  - Late 1980s–90s: Limited data and computing power slow progress → “AI winters”

Prof. Rony G. Flatscher

Machine Learning (ML), 2
Modern Machine Learning (2000s–Present)

- Key Drivers
  - Explosion of data (internet, sensors, mobile devices)
  - Advances in computing power (GPUs, cloud computing)
  - Improved algorithms and optimization techniques
- Major Breakthroughs
  - 2006–2012: Deep learning resurgence
  - 2012: AlexNet revolutionizes image recognition
  - 2010s–Present:
    - Applications in vision, speech, natural language processing (NLP), medicine, and autonomous systems
    - Rise of foundation models and large-scale neural networks
- Today
  - ML is a core technology behind search engines, recommendations, generative AI, and scientific discovery

Prof. Rony G. Flatscher

Machine Learning (ML), 3

- What is Machine Learning ?
  - A field of AI (artificial intelligence) where computers learn patterns from data instead of being explicitly programmed
  - Models improve their performance through experience (data)
  - Used for tasks like prediction, classification, and decision-making
- Why use Machine Learning?
  - Handles complex patterns that are hard to code manually
  - Scales to large and growing datasets
  - Adapts and improves as new data becomes available
  - Enables practical applications such as:
    - Recommendation systems
    - Image and speech recognition
    - Fraud detection and forecasting

Prof. Rony G. Flatscher

Machine Learning (ML), 4

- Building blocks
  - Data
    - Examples the system learns from
  - Model
    - Mathematical representation of patterns
  - Training/learning
    - Adjusting the model using data
  - Prediction
    - Applying learned patterns to new data

Prof. Rony G. Flatscher

Machine Learning (ML), 5

- Main types of machine learning
  - Supervised Learning
    - Uses labeled data
    - Example: spam detection
  - Unsupervised Learning
    - No labels provided, find patterns
    - Example: customer clustering
  - Reinforcement Learning
    - Learns by trial and error
    - Example: game-playing AI

Prof. Rony G. Flatscher

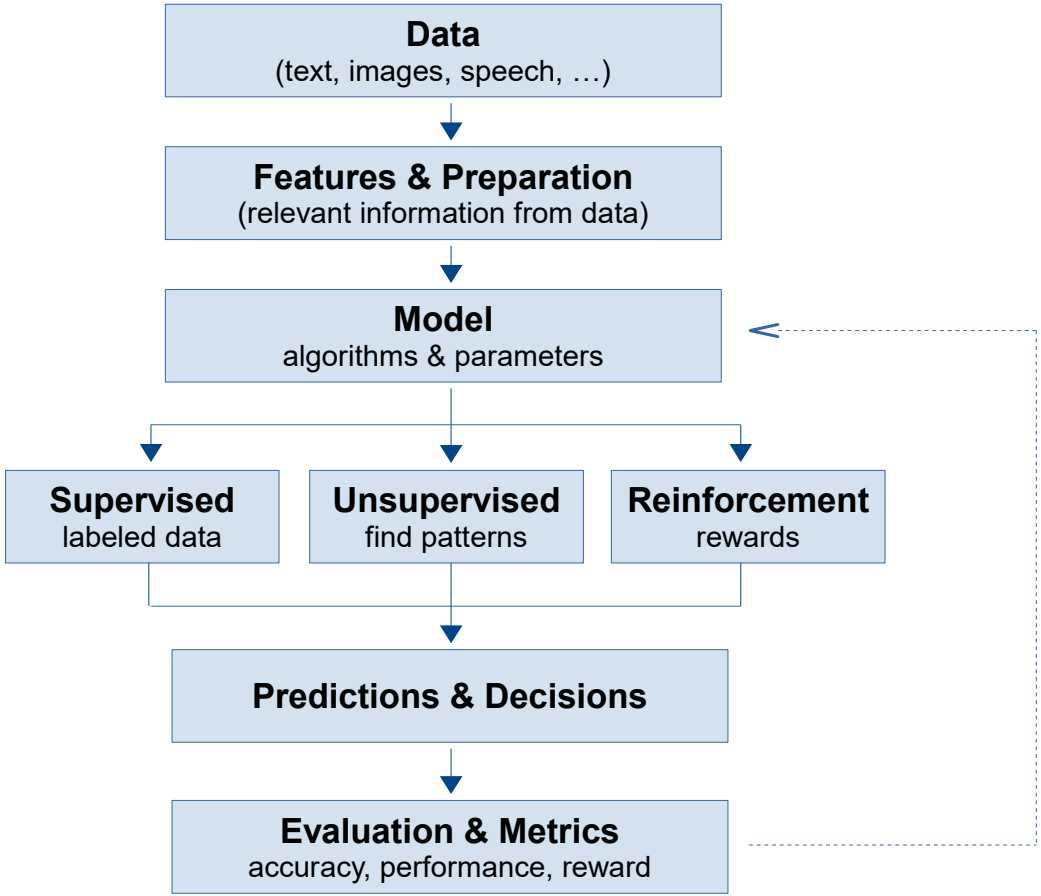
Machine Learning (ML), 6

- Key Components & Challenges
  - Data
    - Text, logs, manuals, images, speech, movies, ...
  - Features and preparation
    - Relevant information from data
  - Model
    - Algorithms and parameters
      - Methods used to learn patterns (supervised, unsupervised, reinforcement, ...)
  - Predictions and decisions
    - Apply model to data
  - Evaluation
    - Measuring accuracy and performance

Challenges: data quality, bias, overfitting, interpretability

Prof. Rony G. Flatscher

Machine Learning (ML), 7



Prof. Rony G. Flatscher

Weka, 1
Waikato Environment for Knowledge Analysis

- Weka (Waikato Environment for Knowledge Analysis) is an open-source machine learning tool developed in New Zealand
  - Created and maintained at the *University of Waikato, New Zealand*
    - https://ml.cms.waikato.ac.nz/weka
  - Java-based
    - Java development started 1997
      - In the 90's development started out with Tcl/Tk, C, ...
      - As of 2025/26 the release version of Weka is 3.8.6, released https://waikato.github.io/weka-wiki/downloading_weka/
    - Runs on Windows, macOS, and Linux
  - Provides an easy-to-use graphical user interface (GUI)
  - Supports classification, regression, clustering, and data visualization, ...
  - Widely used for teaching, research, and data mining

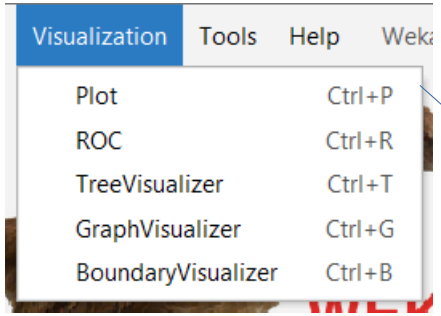
Weka, 2
Core Interfaces

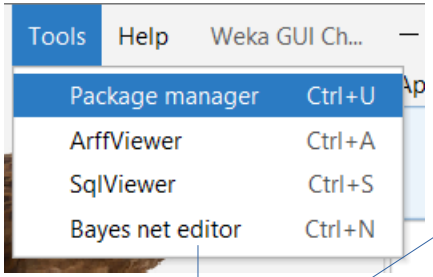
- Explorer (GUI)
  - Main graphical user interface
  - Used for data preprocessing, classification, regression, clustering, and visualization
  - Step-by-step workflow for experimenting with datasets
- Knowledge Flow (GUI)
  - Visual, pipeline-based interface
  - Allows users to design machine learning workflows using connected components
  - Useful for understanding data flow and automating experiments
- Experimenter (GUI)
  - Designed for running and comparing multiple machine learning algorithms
  - Supports statistical evaluation of results
  - Useful for benchmarking and research experiments

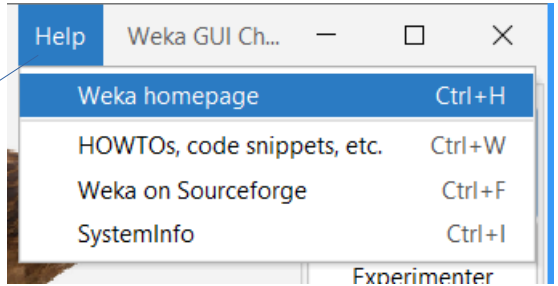
Weka, 3
Core Interfaces

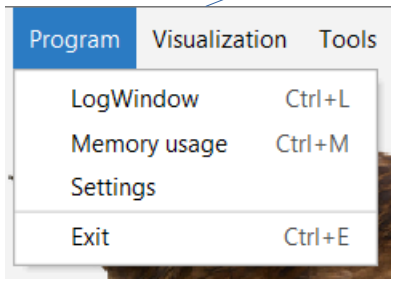
- Workbench (GUI)
  - Integrates multiple Weka tools in one environment
    - Provides access to Explorer, Experimenter, and Knowledge Flow
  - Ideal for advanced users managing multiple tasks
- Command-Line Interface (CLI)
  - Enables scripting and automation
  - Useful for large-scale experiments and batch processing
- Libraries & Algorithms (Java class libraries)
  - Collection of machine learning algorithms (e.g., decision trees, Naïve Bayes, SVMs, clustering methods)
  - Tools for data preprocessing, feature selection, and evaluation
  - ooRexx can be used as a tool and scripting language

Weka, 4
GUI – Main Interface (Command: `java -jar weka.jar`)

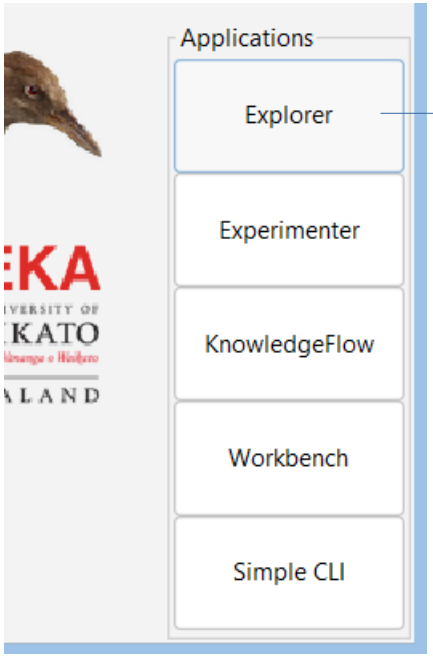


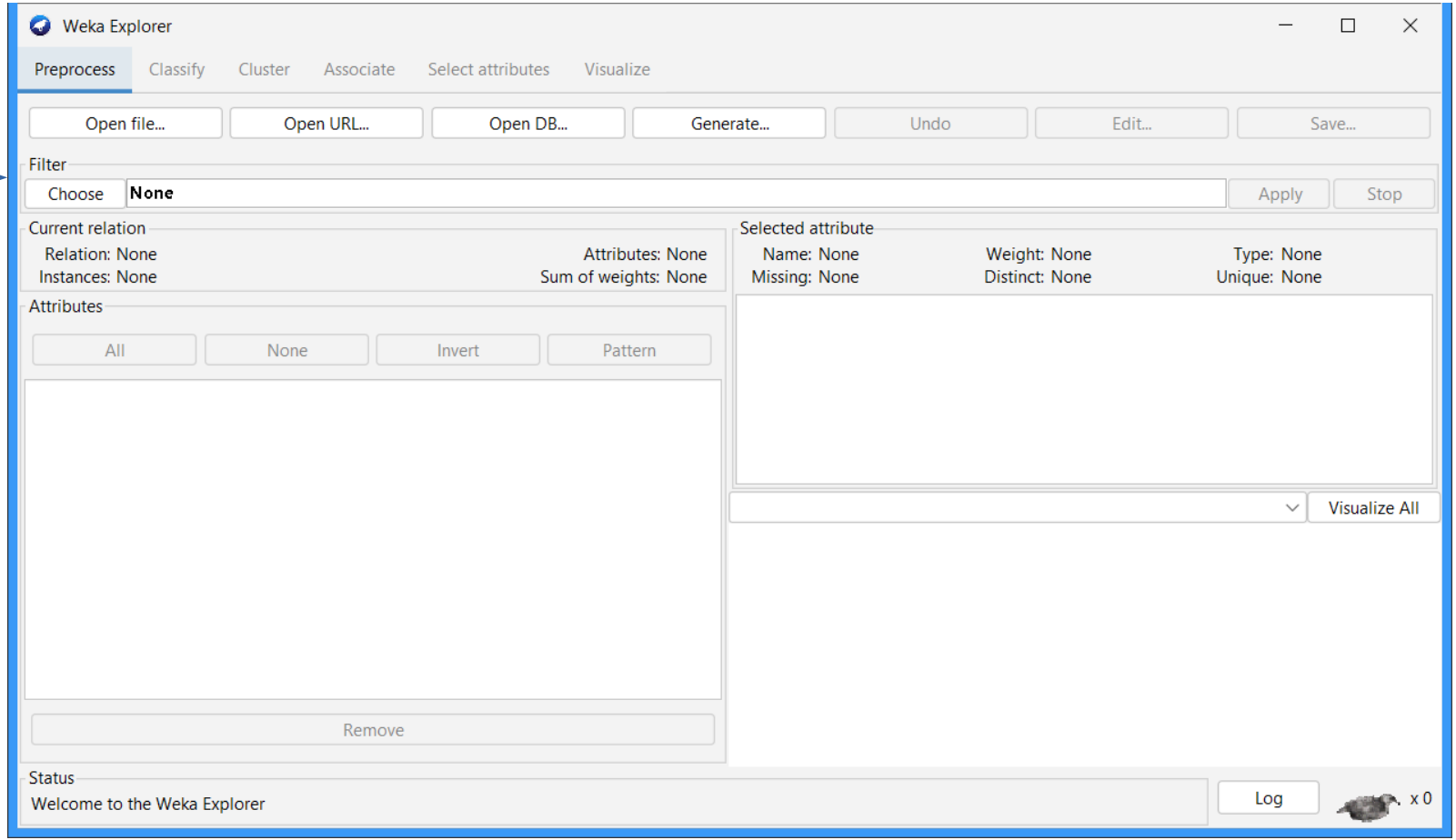






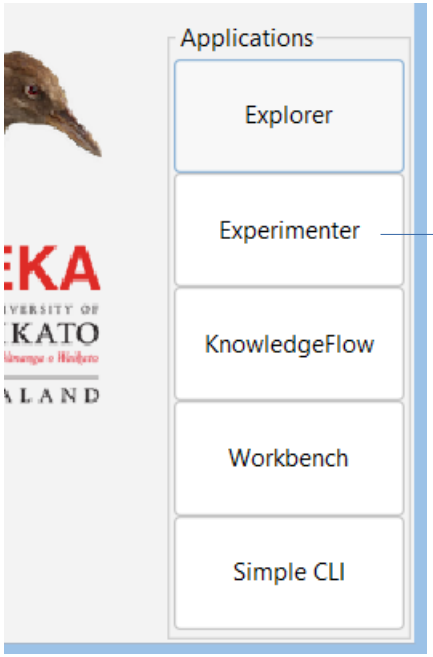
Weka, 5
GUI – Explorer

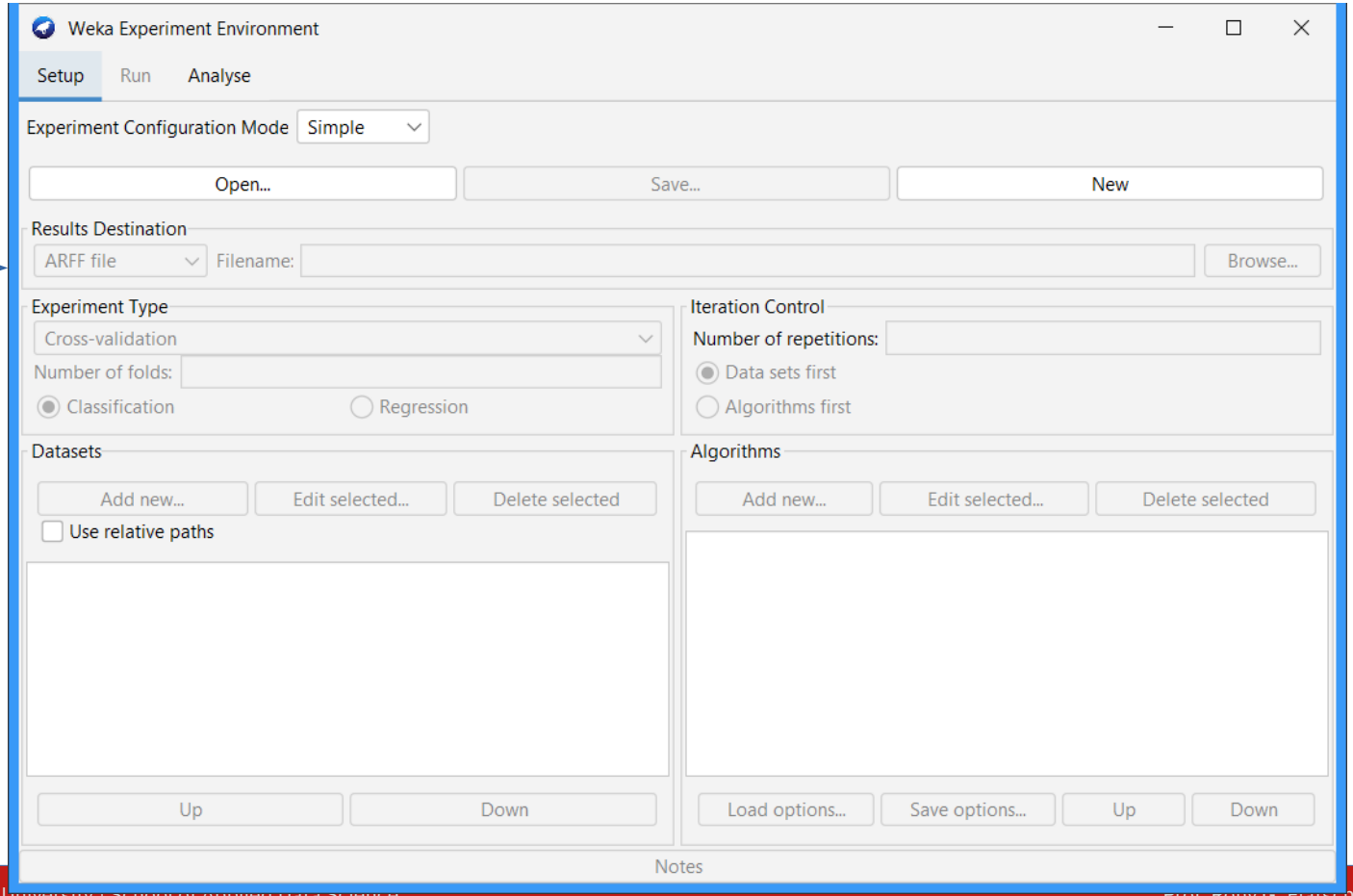




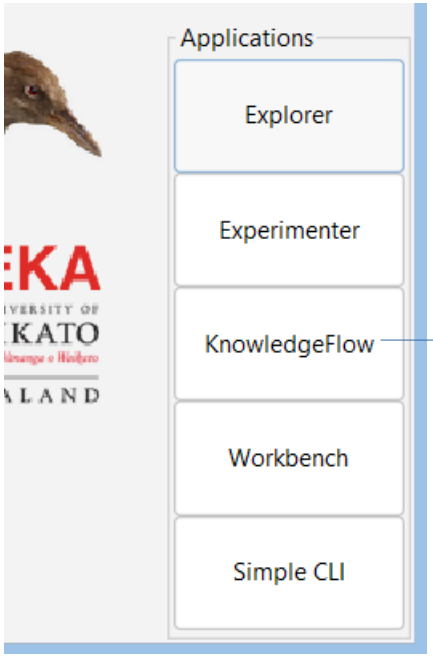
Prof. Rony G. Flatscher

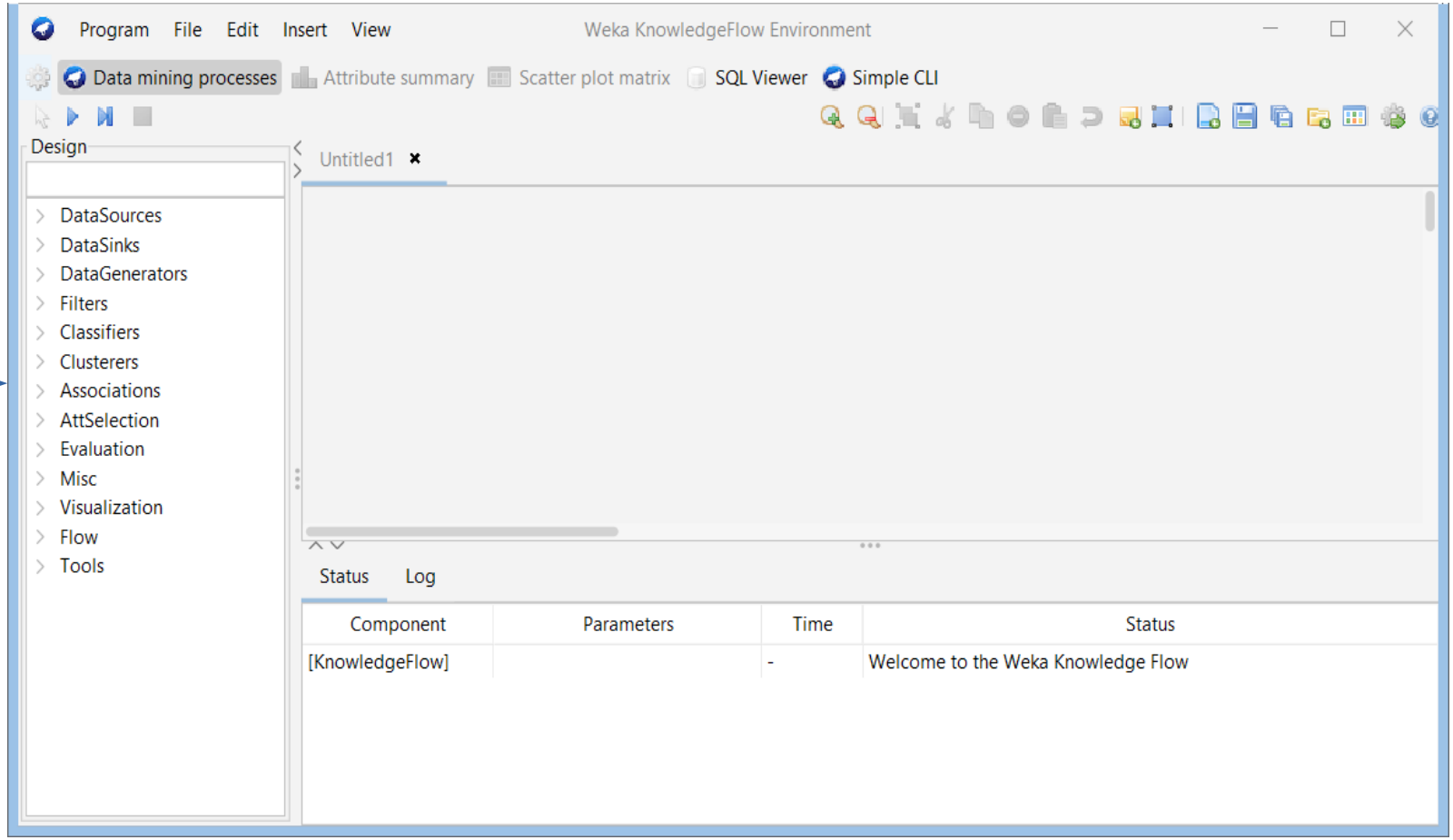
Weka, 6
GUI – Experimenter



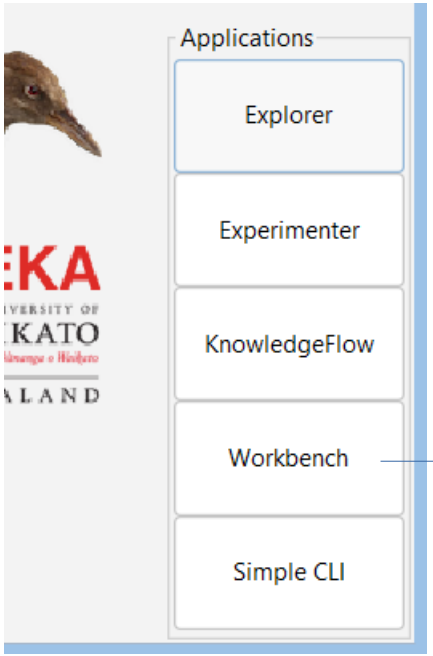


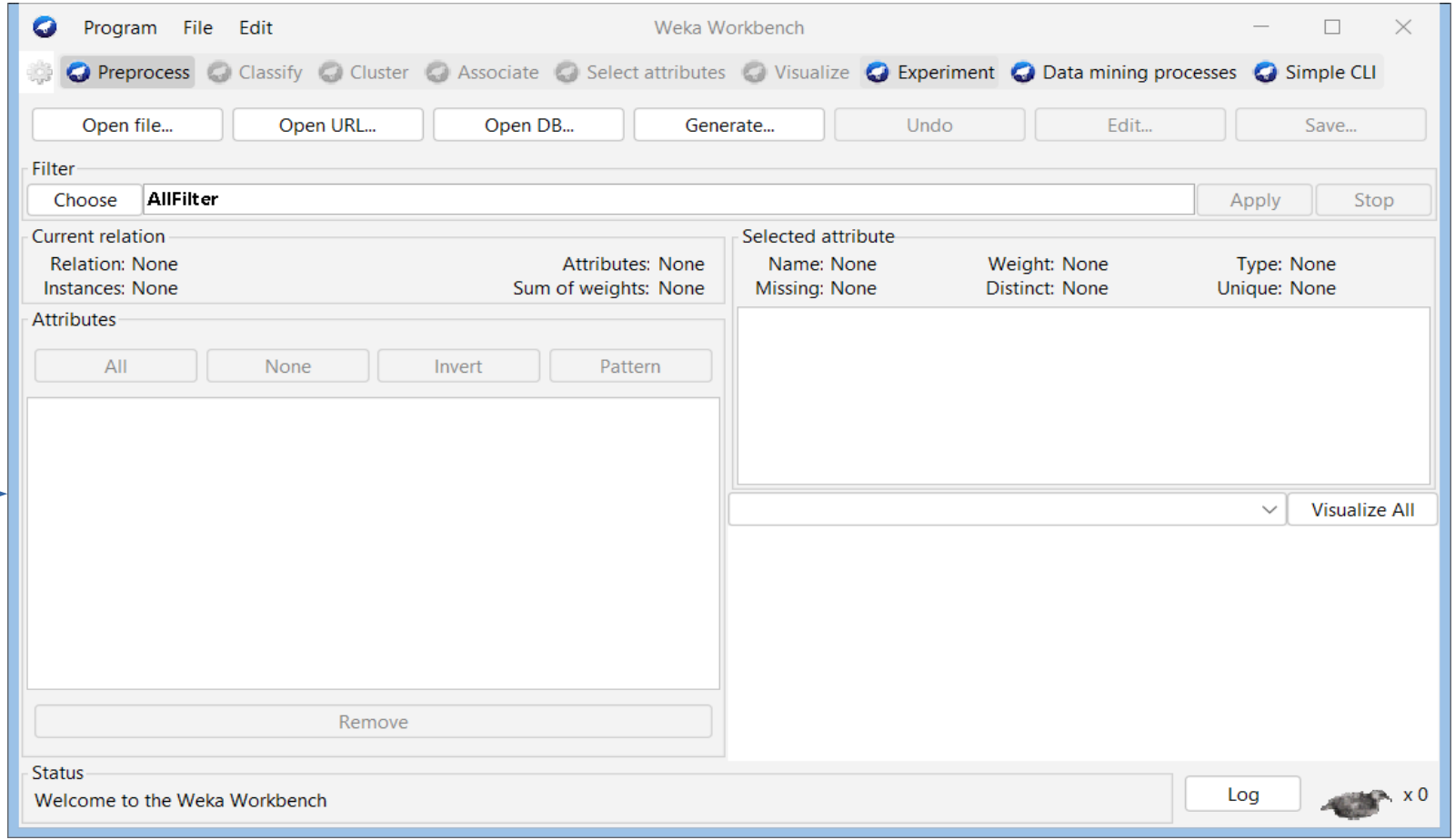
Weka, 7
GUI – KnowledgeFlow





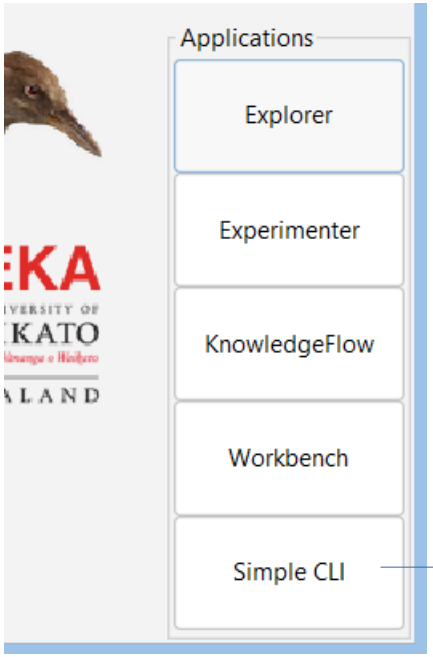
Weka, 8
GUI – Workbench





Prof. Rony G. Flatscher

Weka, 9
GUI – SimpleCLI_1

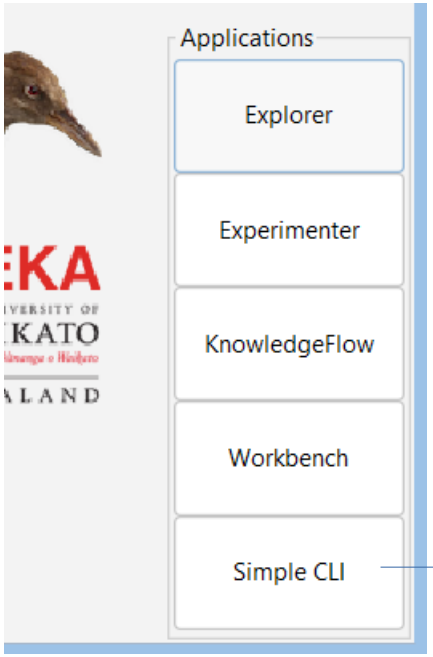


SimpleCLI

```
Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of
the window. Use the up and down arrows to move
through previous commands.
Command completion for classnames and files is
initiated with <Tab>. In order to distinguish
between files and classnames, file names must
be either absolute or start with '.\' or '~/'
(the latter is a shortcut for the home directory).
<Alt+BackSpace> is used for deleting the text
in the commandline in chunks.

Type 'help' followed by <Enter> to see an overview
of all commands.
```

```
help
```

Weka, 10
GUI – SimpleCLI_2



SimpleCLI

```
of all commands.
> help

capabilities <classname> <args>
          Lists the capabilities of the specified class.
          If the class is a weka.core.OptionHandler then
          trailing options after the classname will be
          set as well.

cls
          Clears the output area.

echo msg
          Outputs a message.

exit
          Exits the SimpleCLI program.

help [command1] [command2] [...]
          Outputs the help for the specified command or, if omitted,
          for all commands.

history
          Prints all issued commands.

java <classname> <args>
          Lists the capabilities of the specified class.
          If the class is a weka.core.OptionHandler then
          trailing options after the classname will be
```

Weka's "hello world" Example "Iris"

- Dataset "Iris"
  - Data in text file named `iris.arff`
  - Instances: 150 flower samples
  - Classes: 3 species – Iris setosa, Iris versicolor, Iris virginica (50 each)
  - Attributes (numeric)
    - Sepal length, sepal width, petal length, petal width
  - Ideal for experimenting with classification algorithms, visualization, and evaluation
    - Many classifiers achieve 95–100% accuracy
    - Setosa is linearly separable; versicolor and virginica partially overlap

Iris setosa – Borsten-Schwertlilie (de)
Iris versicolor – Vielfarbige Schwertlilie (de)
Iris virginica – Blaue Sumpfschwertlilie (de)

Sepal – Kelchblatt (de)
Petal – Blütenblatt (de)

- Some tutorials relating to Iris
  - https://deeplearning.cms.waikato.ac.nz/user-guide/getting-started/
  - https://deeplearning.cms.waikato.ac.nz/examples/classifying-iris/
  - https://machinelearningmastery.com/how-to-run-your-first-classifier-in-weka/
  - https://www.geeksforgeeks.org/machine-learning/hierarchical-clustering-using-weka/

Weka's "iris.arff", 1/2

$WEKA_HOME/data/iris.arff or $WEKA_HOME/packages/wekaDeeplearning4j/datasets/nominal/iris.arff

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
% 3. Past Usage:
%    - Publications: too many to mention!!!  Here are a few.
%    1. Fisher,R.A. "The use of multiple measurements in taxonomic prot
%       Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contribut
%       to Mathematical Statistics" (John Wiley, NY, 1950).
%    2. Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene
%       (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 21
%    3. Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New S
%       Structure and Classification Rule for Recognition in Partially
%       Environments".  IEEE Transactions on Pattern Analysis and Machi
%       Intelligence, Vol. PAMI-2, No. 1, 67-71.
%       -- Results:
%          -- very low misclassification rates (0% for the setosa class
%    4. Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE
%       Transactions on Information Theory, May 1972, 431-433.
%       -- Results:
%          -- very low misclassification rates again
%    5. See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al's AUTOC
%       conceptual clustering system finds 3 classes in the data.
%
% 4. Relevant Information:
%    --- This is perhaps the best known database to be found in the pat
%        recognition literature.  Fisher's paper is a classic in the fi
%        and is referenced frequently to this day.  (See Duda & Hart, t
%        example.)  The data set contains 3 classes of 50 instances eac
%        where each class refers to a type of iris plant.  One class is
%        linearly separable from the other 2; the latter are NOT linear
%        separable from each other.
%    --- Predicted attribute: class of iris plant.
%    --- This is an exceedingly simple domain.
%
% 5. Number of Instances: 150 (50 in each of three classes)
%
% 6. Number of Attributes: 4 numeric, predictive attributes and the class
%
% 7. Attribute Information:
%    1. sepal length in cm
%    2. sepal width in cm
%    3. petal length in cm
%    4. petal width in cm
%    5. class:
%       -- Iris Setosa
%       -- Iris Versicolour
%       -- Iris Virginica
%
% 8. Missing Attribute Values: None
%
% Summary Statistics:
%                 Min  Max   Mean    SD   Class Correlation
%    sepal length: 4.3  7.9   5.84  0.83    0.7826
%     sepal width: 2.0  4.4   3.05  0.43   -0.4194
%    petal length: 1.0  6.9   3.76  1.76    0.9490  (high!)
%     petal width: 0.1  2.5   1.20  0.76    0.9565  (high!)
%
% 9. Class Distribution: 33.3% for each of 3 classes.

@RELATION iris

@ATTRIBUTE sepallength  REAL
@ATTRIBUTE sepalwidth   REAL
@ATTRIBUTE petallength  REAL
@ATTRIBUTE petalwidth   REAL
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
```

Weka's "iris.arff", 2/2

$WEKA_HOME/data/iris.arff or $WEKA_HOME/packages/wekaDeeplearning4j/datasets/nominal/iris.arff

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,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
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa
5.2,3.5,1.5,0.2,Iris-setosa
5.2,3.4,1.4,0.2,Iris-setosa
4.7,3.2,1.6,0.2,Iris-setosa
4.8,3.1,1.6,0.2,Iris-setosa
5.4,3.4,1.5,0.4,Iris-setosa
5.2,4.1,1.5,0.1,Iris-setosa
5.5,4.2,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.0,3.2,1.2,0.2,Iris-setosa
```

… cut …

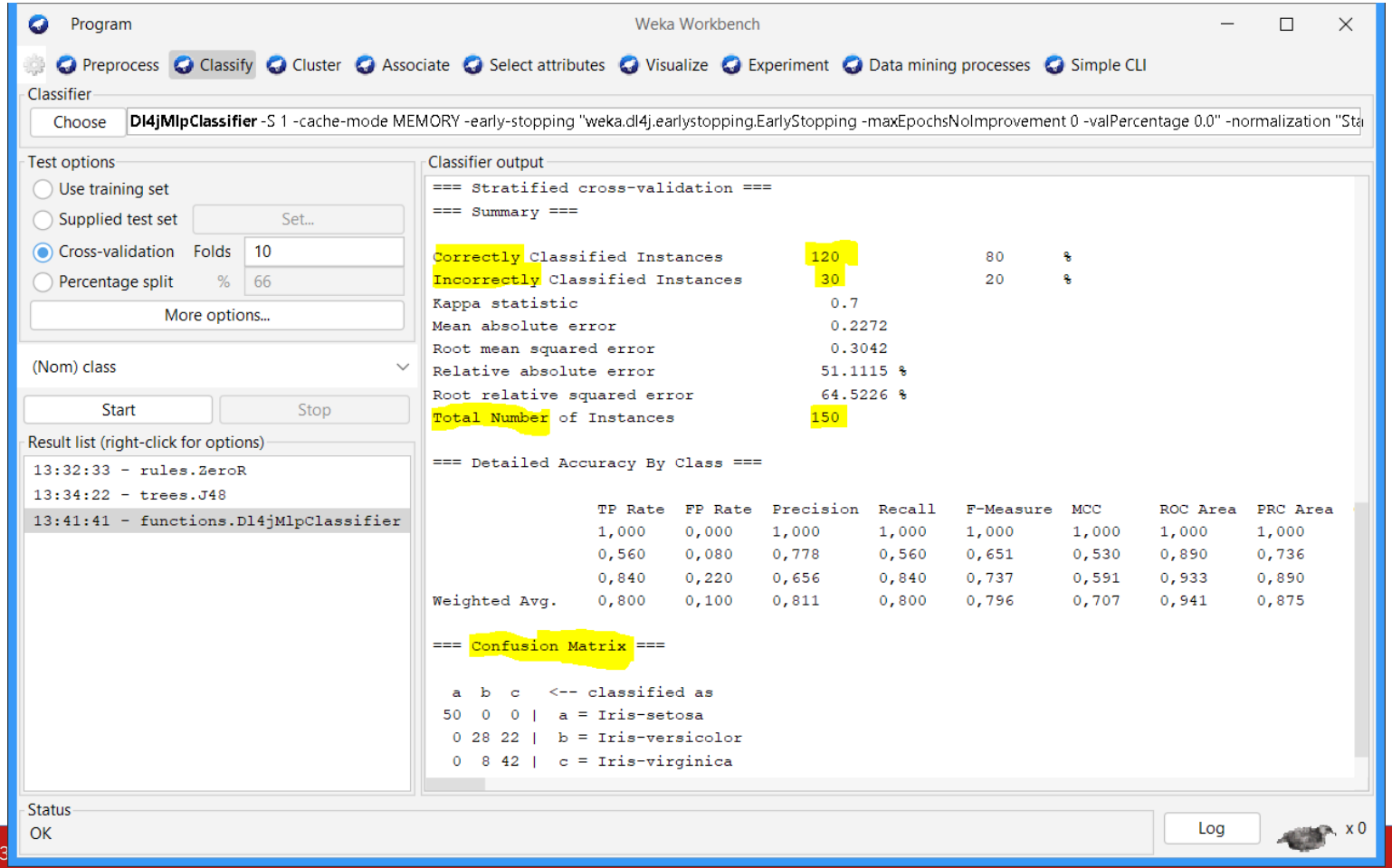
```
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5.0,2.0,3.5,1.0,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor
6.0,2.2,4.0,1.0,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor
6.7,3.1,4.4,1.4,Iris-versicolor
5.6,3.0,4.5,1.5,Iris-versicolor
5.8,2.7,4.1,1.0,Iris-versicolor
6.2,2.2,4.5,1.5,Iris-versicolor
5.6,2.5,3.9,1.1,Iris-versicolor
5.9,3.2,4.8,1.8,Iris-versicolor
6.1,2.8,4.0,1.3,Iris-versicolor
6.3,2.5,4.9,1.5,Iris-versicolor
6.1,2.8,4.7,1.2,Iris-versicolor
6.4,2.9,4.3,1.3,Iris-versicolor
6.6,3.0,4.4,1.4,Iris-versicolor
6.8,2.8,4.8,1.4,Iris-versicolor
6.7,3.0,5.0,1.7,Iris-versicolor
6.0,2.9,4.5,1.5,Iris-versicolor
5.7,2.6,3.5,1.0,Iris-versicolor
5.5,2.4,3.8,1.1,Iris-versicolor
5.5,2.4,3.7,1.0,Iris-versicolor
5.8,2.7,3.9,1.2,Iris-versicolor
```

… cut …

```
6.4,2.8,5.6,2.2,Iris-virginica
6.3,2.8,5.1,1.5,Iris-virginica
6.1,2.6,5.6,1.4,Iris-virginica
7.7,3.0,6.1,2.3,Iris-virginica
6.3,3.4,5.6,2.4,Iris-virginica
6.4,3.1,5.5,1.8,Iris-virginica
6.0,3.0,4.8,1.8,Iris-virginica
6.9,3.1,5.4,2.1,Iris-virginica
6.7,3.1,5.6,2.4,Iris-virginica
6.9,3.1,5.1,2.3,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
6.8,3.2,5.9,2.3,Iris-virginica
6.7,3.3,5.7,2.5,Iris-virginica
6.7,3.0,5.2,2.3,Iris-virginica
6.3,2.5,5.0,1.9,Iris-virginica
6.5,3.0,5.2,2.0,Iris-virginica
6.2,3.4,5.4,2.3,Iris-virginica
5.9,3.0,5.1,1.8,Iris-virginica
%
%
%
~
~
```

Weka Getting Started, GUI Example

https://deeplearning.cms.waikato.ac.nz/user-guide/getting-started/



Weka Getting Started, Java Example

https://deeplearning.cms.waikato.ac.nz/user-guide/getting-started/#java

```java
// https://deeplearning.cms.waikato.ac.nz/user-guide/getting-started/
import weka.classifiers.Evaluation;
import weka.classifiers.functions.Dl4jMlpClassifier;
import weka.core.Instances;

import java.io.FileReader;
import java.nio.file.Paths;
import java.util.Random;

public class TestWeka {
    public static void main(String[] args) throws Exception {
        Dl4jMlpClassifier clf = new Dl4jMlpClassifier();
        String irisPath = Paths.get(System.getenv("WEKA_HOME"),
                          "packages", "wekaDeeplearning4j", "datasets",
                          "nominal", "iris.arff").toString();
        Instances inst = new Instances(new FileReader(irisPath));
        inst.setClassIndex(inst.numAttributes() - 1);
        Evaluation ev = new Evaluation(inst);
        ev.crossValidateModel(clf, inst, 10, new Random(0));
        System.out.println(ev.toSummaryString());
    }
}
```


Prof. Rony G. Flatscher

Weka Getting Started, ooRexx Example

https://deeplearning.cms.waikato.ac.nz/user-guide/getting-started/#java

```
   -- load the classifier function
clf      = .bsf~new("weka.classifiers.functions.Dl4jMlpClassifier")
   -- get path to WEKA home directory from enivronment, define path to data
   -- (forward slash works on Unix and on Windows)
irisPath = value("WEKA_HOME",,"environment")"/data/iris.arff"
   -- read Iris data
fReader  = .bsf~new("java.io.FileReader",irisPath)
inst     = .bsf~new("weka.core.Instances",fReader)
inst~setClassIndex(inst~numAttributes - 1)
   -- create evaluation for Iris data
ev       = .bsf~new("weka.classifiers.Evaluation", inst)
   -- cross Validate
ev~crossValidateModel(clf, inst, 10, .bsf~new("java.util.Random",0))
   -- show summary as a string
say ev~toSummaryString

::requires "BSF.CLS" -- get ooRexx-Java bridge
```


Prof. Rony G. Flatscher

Weka Getting Started, ooRexx Example

Output of running the Java or ooRexx Program

```
Administrator: C:\WINDOWS\system32\cmd.exe
Jän. 18, 2026 12:34:55 PM com.github.fommil.jni.JniLoader liberalLoad
INFO: successfully loaded C:\Users\ADMINI~1\AppData\Local\Temp\jniloader1320553152165553819netlib-native_ref-win-x86_64.dll
[INFO ] 12:34:56.151 [main] weka.classifiers.functions.Dl4jMlpClassifier - Building on 135 training instances
[INFO ] 12:34:56.194 [main] org.nd4j.linalg.factory.Nd4jBackend - Loaded [CpuBackend] backend
[INFO ] 12:34:56.991 [main] org.nd4j.nativeblas.NativeOpsHolder - Number of threads used for linear algebra: 4
[WARN ] 12:34:56.995 [main] org.nd4j.linalg.cpu.nativecpu.CpuNDArrayFactory - ********************************* CPU Feature Check Warning *********************************
[WARN ] 12:34:56.998 [main] org.nd4j.linalg.cpu.nativecpu.CpuNDArrayFactory - Warning: Initializing ND4J with Generic x86 binary on a CPU with AVX/AVX2 support
[WARN ] 12:34:57.000 [main] org.nd4j.linalg.cpu.nativecpu.CpuNDArrayFactory - Using ND4J with AVX/AVX2 will improve performance. See deeplearning4j.org/cpu for more details
[WARN ] 12:34:57.003 [main] org.nd4j.linalg.cpu.nativecpu.CpuNDArrayFactory - Or set environment variable ND4J_IGNORE_AVX=true to suppress this warning
[WARN ] 12:34:57.005 [main] org.nd4j.linalg.cpu.nativecpu.CpuNDArrayFactory - *********************************************************************************************
[INFO ] 12:34:57.105 [main] org.nd4j.nativeblas.Nd4jBlas - Number of threads used for OpenMP BLAS: 4
[INFO ] 12:34:57.128 [main] org.nd4j.linalg.api.ops.executioner.DefaultOpExecutioner - Backend used: [CPU]; OS: [Windows 10]
[INFO ] 12:34:57.130 [main] org.nd4j.linalg.api.ops.executioner.DefaultOpExecutioner - Cores: [8]; Memory: [5,9GB];
[INFO ] 12:34:57.134 [main] org.nd4j.linalg.api.ops.executioner.DefaultOpExecutioner - Blas vendor: [OPENBLAS]
WARNING: A terminally deprecated method in sun.misc.Unsafe
WARNING: sun.misc.Unsafe::arrayBaseOffset has been called
WARNING: Please consider reporting this to the maintainers
WARNING: sun.misc.Unsafe::arrayBaseOffset will be removed
[INFO ] 12:34:57.295 [main] org.deeplearning4j.nn.graph.Co
Backend is: org.nd4j.linalg.cpu.nativecpu.cpubackend



Training Dl4jMlpClassifier...: [] ETA: 00:00:00[INFO ] 12:
Training Dl4jMlpClassifier...: [======
k 00:00:00.128
Training Dl4jMlpClassifier...: [============
k 00:00:00.101
Training Dl4jMlpClassifier...: [==================
k 00:00:00.106
Training Dl4jMlpClassifier...: [========================
Train Set:
 Loss:            0,393745


[INFO ] 12:34:58.345 [main] weka.classifiers.functions.Dl4
Training Dl4jMlpClassifier...: [==========================
k 00:00:00.093
Training Dl4jMlpClassifier...: [==========================
```

```
Administrator: C:\WINDOWS\system32\cmd.exe
Training Dl4jMlpClassifier...: [======                                                    ] ETA: 00:00:00[INFO ] 12:35:05.323 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [2/
k 00:00:00.040
Training Dl4jMlpClassifier...: [============                                              ] ETA: 00:00:00[INFO ] 12:35:05.372 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [3/
k 00:00:00.042
Training Dl4jMlpClassifier...: [==================                                        ] ETA: 00:00:00[INFO ] 12:35:05.439 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [4/
k 00:00:00.057
Training Dl4jMlpClassifier...: [========================                                  ] ETA: 00:00:00[INFO ] 12:35:05.536 [main] weka.dl4j.listener.EpochListener - Epoch [5/10]
Train Set:
 Loss:            0,271228


[INFO ] 12:35:05.541 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [5/10] took 00:00:00.093
Training Dl4jMlpClassifier...: [==============================                            ] ETA: 00:00:00[INFO ] 12:35:05.590 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [6/
k 00:00:00.048
Training Dl4jMlpClassifier...: [==================================                        ] ETA: 00:00:00[INFO ] 12:35:05.658 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [7/
k 00:00:00.064
Training Dl4jMlpClassifier...: [========================================                  ] ETA: 00:00:00[INFO ] 12:35:05.740 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [8/
k 00:00:00.074
Training Dl4jMlpClassifier...: [==============================================            ] ETA: 00:00:00[INFO ] 12:35:05.801 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [9/
k 00:00:00.056
Training Dl4jMlpClassifier...: [====================================================      ] ETA: 00:00:00[INFO ] 12:35:05.933 [main] weka.dl4j.listener.EpochListener - Epoch [10/10]
Train Set:
 Loss:            0,088797


[INFO ] 12:35:05.949 [main] weka.classifiers.functions.Dl4jMlpClassifier - Epoch [10/10] took 00:00:00.135
Training Dl4jMlpClassifier...: [============================================================] ETA: 00:00:00
Done!


Correctly Classified Instances         120               80      %
Incorrectly Classified Instances        30               20      %
Kappa statistic                          0.7
Mean absolute error                      0.2268
Root mean squared error                  0.3031
Relative absolute error                 51.0306 %
Root relative squared error             64.3032 %
Total Number of Instances              150
```

Weka – Nutshell Examples in ooRexx

- Based on a seminar paper with nutshell examples by the WU student Jakov Pavic
  - Useful resources homepage: https://wi.wu.ac.at/rgf/diplomarbeiten/#sem_202307_01
    - Seminar paper with a brief, but very helpful introduction to Weka
      https://wi.wu.ac.at/rgf/diplomarbeiten/Seminararbeiten/2023/202307_PavicJakov_WEKA.pdf
    - Java nutshell examples:
      https://wi.wu.ac.at/rgf/diplomarbeiten/Seminararbeiten/2023/202307_PavicJakov_WEKA-data.zip
- Weka, seminar paper, nutshell examples in

  https://wi.wu.ac.at/rgf/wu/lehre/autojava/material/foils/370_Weka_20260122_rgf_WU.zip
  - Includes updated weka.jar, seminar paper, all nutshell examples
  - After downloading and unzipping study `readme.txt`

Weka – `setupWeka.cmd/setupWeka.sh`

- Open a terminal, change into the unzipped folder and run
  - Windows: `setupWeka`
  - Unix: `source sh ./setupWeka.sh`
  - To run the Weka GUI: `java -jar weka.jar`
  - To run the ooRexx examples use the `rexxjh.cmd`/`rexxj.sh` launchers installed with BSF4ooRexx (Java-ooRexx bridge), e.g.
    - Windows: `rexxjh WU\01_TestWeka\rxTestWeka.rxj`
    - Unix: `rexxjh.sh WU/01_TestWeka/rxTestWeka.rxj`
- The seminar paper and all its Java and corresponding ooRexx nutshell examples can be found in subdirectory WU, subdirectories `01_TestWeka`, and `02_PavicJacov\nutshells_examples`

Weka – Process and Prepare Data

- It may be necessary to pre-process the data for various reasons
- In this example we use three different files with data
  - One containing the initial data: `housing.csv`
  - One containing two columns to be merged to the initial data: `housing_newColumns.arff`
  - One containing new data to be added: `housing_newRows.csv`
- We get various information from the data like mean and mode
- We filter rows by a certain price and remove duplicates
- We use a filter to add a new attribute (`price_per_sqm`)
- Show how to split the data into a training and testing section
- Save the prepared data as `arff` and as `csv`

Weka – Process and Prepare Data

Initial Data

housing.csv

```
city,country,type,sqr_metre,price
London,United Kingdom,Apartment,?,200000
Paris,France,House,120,350000
Berlin,Germany,Apartment,60,150000
Madrid,Spain,Apartment,90,180000
Rome,Italy,House,150,400000
Amsterdam,Netherlands,Apartment,70,250000
Vienna,Austria,House,110,300000
Athens,Greece,Apartment,75,160000
Stockholm,Sweden,Apartment,85,220000
Dublin,Ireland,House,130,380000
~
~
```

housing_new_columns.arff

```
@relation housing_newColumns

@attribute ROOMS numeric
@attribute balcony_y/n {y,n}

@data
2,y^M
4,n^M
1,n^M
3,y^M
5,y^M
2,n^M
4,y^M
2,n^M
3,y^M
4,y^M
~
~
```

housing_newRows.csv

```
city,country,type,sqr_metre,price,ROOMS,balcony_y/n
Paris,France,House,120,350000, 4,n
Barcelona,Spain,House,160,420000,6,n
Brussels,Belgium,Apartment,65,180000,2,n
Lisbon,Portugal,Apartment,95,200000,3,y
Prague,Czech Republic,?,100,280000,4,y
~
~
```

Weka – Process and Prepare Data, 1/4

WU\02_PavicJacov\nutshells_examples\data_preparation\data_preparation.rxj

```
/* ooRexx version of "data_preparation.java" */
   -- abstract Java class, cannot be imported, hence bsf.loadClass
call bsf.loadClass "weka.filters.Filter", Filter
   -- import Java classes into ooRexx
      -- DataSource class is embedded in ConverterUtils
call bsf.import "weka.core.converters.ConverterUtils$DataSource", DataSource
call bsf.import "weka.core.Instances", Instances
call bsf.import "java.io.File", "ioFile"

-- IMPORTING + COMBINING DATA
housing        = .DataSource~new("./housing.csv")~getDataSet
housing_newcol = .DataSource~new("./housing_newColumns.arff")~getDataSet
housing_new    = .Instances~mergeInstances(housing,housing_newcol)
   -- get new rows and add them in a loop
housing_newrow = .DataSource~new("./housing_newRows.csv")~getDataSet
do i=0 to housing_newrow~numInstances-1
   housing_new~add(housing_newrow~instance(i))
end

-- UNDERSTANDING DATA
do i=0 to 4 -- show the first five instances
   say housing_new~get(i) ~toString
end
/* or iterating using ooRexx' loop...over
   loop data over housing_new for 5
      say data~toString
   end
*/

say housing_new~toSummaryString  -- show summary of the data

say "housing_new:"
say housing_new~toString --  … continued …
```


Prof. Rony G. Flatscher

Weka – Process and Prepare Data, 2/4

WU\02_PavicJacov\nutshells_examples\data_preparation\data_preparation.rxj

```
say "housing_new~numAttributes:" housing_new~numAttributes
do i=0 to housing_new~numAttributes-1
   attr=housing_new~attribute(box('int',i))  -- force use of Instances.attribute(int)
end

do i=0 to housing_new~numAttributes-1  -- mean/mode for each attribute
   attr=housing_new~attribute(box('int',i))  -- force use of Instances.attribute(int)
   if attr~type = 0 then
      say "The mean of the" attr~name "attribute is:" housing_new~meanOrMode(i)
   else
   do
      index = format(housing_new~meanOrMode(i),,0)  --get the index of the mode value
      say "The mode of the" attr~name "attribute is:" attr~value(index)
   end
end

numInst=housing_new~numInstances
do i=0 to housing_new~numAttributes-1  -- min/max for numeric attributes
   attr=housing_new~attribute(box('int',i))  -- force use of Instances.attribute(int)
   if attr~type = 0 then
   do
      say "The smallest value of the" attr~name "variable is:" housing_new~kthSmallestValue(i, 1)
      say "The biggest value of the"  attr~name "variable is:" housing_new~kthSmallestValue(i, numInst)
   end
end

--SELECTING DATA
--Delete unnecessary Attribute (balcony_y/n)
housing_new~deleteAttributeAt(6) -- 0-based, hence 7th attribute

-... continued …
```

Weka – Process and Prepare Data, 3/4

WU\02_PavicJacov\nutshells_examples\data_preparation\data_preparation.rxj

```
--Filter every value below 170000
filter_min = .bsf~new("weka.filters.unsupervised.instance.RemoveWithValues")
options = bsf.createJavaArrayOf("java.lang.String", /* column */ "-C", 5, /* smaller */ "-S", 170000)
filter_min~setOptions(options)
filter_min~setInputFormat(housing_new)
housing_expensive = .Filter~useFilter(housing_new, filter_min)

--CLEANING DATA
--Duplicate Detection
filter = .bsf~new("weka.filters.unsupervised.instance.RemoveDuplicates")
filter~setInputFormat(housing_expensive)
housing_nodup = .Filter~useFilter(housing_expensive, filter)

--Rename Attribute
housing_nodup~renameAttribute(5,"rooms")   --rename column 5 to rooms

--Remove rows where attributes miss values
say "housing_nodup:"
say housing_nodup~toString
do i=housing_nodup~numInstances-1 to 0 by -1
   if housing_nodup~instance(i)~hasMissingValue then housing_nodup~remove(i)
end
say "(AFTER removing) housing_nodup:"
say housing_nodup~toString

--CREATE NEW DATA (create new column)
addExpressionFilter=.bsf~new("weka.filters.unsupervised.attribute.AddExpression")
addExpressionFilter~setExpression("a5 / a4")       --how should the new column look like (fifth column divided by forth column)
addExpressionFilter~setName("price_per_sqrm")      --set the name of the new column
addExpressionFilter~setInputFormat(housing_nodup)  --set input for Filter (data set)
housing_prepared = .Filter~useFilter(housing_nodup, addExpressionFilter)   --use the filter
-... continued …
```


Prof. Rony G. Flatscher

Weka – Process and Prepare Data, 4/4

WU\02_PavicJacov\nutshells_examples\data_preparation\data_preparation.rxj

```
--SPLITTING THE DATA
seed = 42   --set seed for randomization
random = .bsf~new("java.util.Random", seed)
trainPercentage = 80      --set percentage size of training data
housing~randomize(random) --randomize data with seed
trainSize = format(housing~numInstances * trainPercentage / 100, ,0)   --calculate training data size
testSize  = housing~numInstances - trainSize       --calculate testing data size
trainData = .Instances~new(housing, 0, trainSize)  --create training data
testData  = .Instances~new(housing, trainSize, testSize) --create testing data

--SAVING DATA
sv_arff = .bsf~new("weka.core.converters.ArffSaver")
sv_arff~~setInstances(housing_prepared) ~~setFile(.ioFile~new("./housing_prepared.arff")) ~~writeBatch
sv_csv = .bsf~new("weka.core.converters.CSVSaver")   --create saving instance
sv_csv~~setInstances(housing_prepared) ~~setFile(.ioFile~new("./housing_prepared.csv")) ~~writeBatch

::requires "BSF.CLS" -- get Java-ooRexx bridge
```


Prof. Rony G. Flatscher

Weka – Process and Prepare Data

Program Output 1/3

```
command: "java"   org.rexxla.bsf.RexxDispatcher data_preparation.rxj
NOTE: Picked up JDK_JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=jdk.compiler/com.sun.tools.javac.processing=ALL-UNNAMED
Jän. 23, 2026 12:12:02 AM com.github.fommil.netlib.ARPACK <clinit>
WARNING: Failed to load implementation from: com.github.fommil.netlib.NativeSystemARPACK
Jän. 23, 2026 12:12:02 AM com.github.fommil.jni.JniLoader liberalLoad
INFO: successfully loaded C:\Users\ADMINI~1\AppData\Local\Temp\jniloader8408960900820874739netlib-native_ref-win-x86_64.dll
London,'United Kingdom',Apartment,?,200000,2,y
Paris,France,House,120,350000,4,n
Berlin,Germany,Apartment,60,150000,1,n
Madrid,Spain,Apartment,90,180000,3,y
Rome,Italy,House,150,400000,5,y
Relation Name:  housing_housing_newColumns
Num Instances:  15
Num Attributes: 7

     Name                     Type  Nom  Int Real     Missing      Unique  Dist
1 city                     Nom 100%   0%   0%     0 /  0%     5 / 33%    10
2 country                  Nom 100%   0%   0%     0 /  0%     5 / 33%    10
3 type                     Nom  93%   0%   0%     1 /  7%     0 /  0%     2
4 sqr_metre                Num   0%  93%   0%     1 /  7%    12 / 80%    13
5 price                    Num   0% 100%   0%     0 /  0%     9 / 60%    12
6 ROOMS                    Num   0% 100%   0%     0 /  0%     3 / 20%     6
7 balcony_y/n              Nom 100%   0%   0%     0 /  0%     0 /  0%     2

housing_new:
@relation housing_housing_newColumns

@attribute city {London,Paris,Berlin,Madrid,Rome,Amsterdam,Vienna,Athens,Stockholm,Dublin}
@attribute country {'United Kingdom',France,Germany,Spain,Italy,Netherlands,Austria,Greece,Sweden,Ireland}
@attribute type {Apartment,House}
@attribute sqr_metre numeric
@attribute price numeric
@attribute ROOMS numeric
@attribute balcony_y/n {y,n}

@data
London,'United Kingdom',Apartment,?,200000,2,y
Paris,France,House,120,350000,4,n
Berlin,Germany,Apartment,60,150000,1,n
Madrid,Spain,Apartment,90,180000,3,y
```

Weka – Process and Prepare Data

Program Output 2/3

```
Rome,Italy,House,150,400000,5,y
Amsterdam,Netherlands,Apartment,70,250000,2,n
Vienna,Austria,House,110,300000,4,y
Athens,Greece,Apartment,75,160000,2,n
Stockholm,Sweden,Apartment,85,220000,3,y
Dublin,Ireland,House,130,380000,4,y
London,'United Kingdom',Apartment,120,350000,4,y
Paris,France,Apartment,160,420000,6,y
Berlin,Germany,House,65,180000,2,y
Madrid,Spain,House,95,200000,3,n
Rome,Italy,?,100,280000,4,n
housing_new~numAttributes: 7
The mode of the city attribute is: London
The mode of the country attribute is: United Kingdom
The mode of the type attribute is: Apartment
The mean of the sqr_metre attribute is: 102.14285714285714
The mean of the price attribute is: 268000.0
The mean of the ROOMS attribute is: 3.2666666666666666
The mode of the balcony_y/n attribute is: y
The smallest value of the sqr_metre variable is: 60.0
The biggest value of the sqr_metre variable is: 1.7976931348623157E308
The smallest value of the price variable is: 150000.0
The biggest value of the price variable is: 420000.0
The smallest value of the ROOMS variable is: 1.0
The biggest value of the ROOMS variable is: 6.0
housing_nodup:
@relation housing_housing_newColumns-weka.filters.unsupervised.instance.RemoveWithValues-S170000.0-C5-Lfirst-last-weka.filters.unsupervised.instance.RemoveDuplicates

@attribute city {London,Paris,Berlin,Madrid,Rome,Amsterdam,Vienna,Athens,Stockholm,Dublin}
@attribute country {'United Kingdom',France,Germany,Spain,Italy,Netherlands,Austria,Greece,Sweden,Ireland}
@attribute type {Apartment,House}
@attribute sqr_metre numeric
@attribute price numeric
@attribute rooms numeric
```

Weka – Process and Prepare Data

Program Output 3/3

```
@data
London,'United Kingdom',Apartment,?,200000,2
Paris,France,House,120,350000,4
Madrid,Spain,Apartment,90,180000,3
Rome,Italy,House,150,400000,5Amsterdam,Netherlands,Apartment,70,250000,2
Vienna,Austria,House,110,300000,4
Stockholm,Sweden,Apartment,85,220000,3
Dublin,Ireland,House,130,380000,4
London,'United Kingdom',Apartment,120,350000,4
Paris,France,Apartment,160,420000,6
Berlin,Germany,House,65,180000,2
Madrid,Spain,House,95,200000,3
Rome,Italy,?,100,280000,4
(AFTER removing) housing_nodup:
@relation housing_housing_newColumns-weka.filters.unsupervised.instance.RemoveWithValues-S170000.0-C5-Lfirst-last-weka.filters.unsupervised.instance.RemoveDuplicates

@attribute city {London,Paris,Berlin,Madrid,Rome,Amsterdam,Vienna,Athens,Stockholm,Dublin}
@attribute country {'United Kingdom',France,Germany,Spain,Italy,Netherlands,Austria,Greece,Sweden,Ireland}
@attribute type {Apartment,House}
@attribute sqr_metre numeric
@attribute price numeric
@attribute rooms numeric

@data
Paris,France,House,120,350000,4
Madrid,Spain,Apartment,90,180000,3
Rome,Italy,House,150,400000,5
Amsterdam,Netherlands,Apartment,70,250000,2
Vienna,Austria,House,110,300000,4
Stockholm,Sweden,Apartment,85,220000,3
Dublin,Ireland,House,130,380000,4
London,'United Kingdom',Apartment,120,350000,4
Paris,France,Apartment,160,420000,6
Berlin,Germany,House,65,180000,2
Madrid,Spain,House,95,200000,3
```


Prof. Rony G. Flatscher

Weka – Process and Prepare Data, 3

Prepared Data

housing_prepared.arff

```
@relation 'housing_housing_newColumns-weka.filters.unsupervised.instance.RemoveWithValues-S170000.0-C5-Lfi
rst-last-weka.filters.unsupervised.instance.RemoveDuplicates-weka.filters.unsupervised.attribute.AddExpres
sion-Ea5 / a4-Nprice_per_sqrm'

@attribute city {London,Paris,Berlin,Madrid,Rome,Amsterdam,Vienna,Athens,Stockholm,Dublin}
@attribute country {'United Kingdom',France,Germany,Spain,Italy,Netherlands,Austria,Greece,Sweden,Ireland}
@attribute type {Apartment,House}
@attribute sqr_metre numeric
@attribute price numeric
@attribute rooms numeric
@attribute price_per_sqrm numeric

@data
Paris,France,House,120,350000,4,2916.666667^M
Madrid,Spain,Apartment,90,180000,3,2000^M
Rome,Italy,House,150,400000,5,2666.666667^M
Amsterdam,Netherlands,Apartment,70,250000,2,3571.428571^M
Vienna,Austria,House,110,300000,4,2727.272727^M
Stockholm,Sweden,Apartment,85,220000,3,2588.235294^M
Dublin,Ireland,House,130,380000,4,2923.076923^M
London,'United Kingdom',Apartment,120,350000,4,2916.666667^M
Paris,France,Apartment,160,420000,6,2625^M
Berlin,Germany,House,65,180000,2,2769.230769^M
Madrid,Spain,House,95,200000,3,2105.263158^M
~
```

housing_prepared.csv

```
City,country,type,sqr_metre,price,rooms,price_per_sqrm
Paris,France,House,120,350000,4,2916.666667
Madrid,Spain,Apartment,90,180000,3,2000
Rome,Italy,House,150,400000,5,2666.666667
Amsterdam,Netherlands,Apartment,70,250000,2,3571.428571
Vienna,Austria,House,110,300000,4,2727.272727
Stockholm,Sweden,Apartment,85,220000,3,2588.235294
Dublin,Ireland,House,130,380000,4,2923.076923
London,'United Kingdom',Apartment,120,350000,4,2916.666667
Paris,France,Apartment,160,420000,6,2625
Berlin,Germany,House,65,180000,2,2769.230769
Madrid,Spain,House,95,200000,3,2105.263158
~
```

Weka – Linear Regression

- There is csv data (`house_prices.csv`) about square meters, bed rooms, bath rooms and the price like this

```
SquareMeters,Bedrooms,Bathrooms,Price
139,3,2,250000
167,4,3,350000
111,2,1,180000
186,3,2,400000
204,4,3,380000
158,3,2,320000
130,2,1,220000
149,3,2,260000
177,4,3,410000
…
```

- Create a linear regression model and evaluate it

Prof. Rony G. Flatscher

Weka – Linear Regression

WU\02_PavicJacov\nutshells_examples\supervised_learning\linear_regression.rxj

```
-- DataSource class is embedded in ConverterUtils
call bsf.import "weka.core.converters.ConverterUtils$DataSource", DataSource
call bsf.import "weka.core.Instances", Instances

-- load packages
.bsf~new("weka.core.WekaPackageManager") ~loadPackages(.false) -- .false: quiet
-- import data set
weather = .DataSource~new("./house_prices.csv") ~getDataSet
-- shuffle and split between training and testing data
seed    = 42
weather~randomize( .bsf~new("java.util.Random",seed) )
trainPercentage = 80
trainSize = format(weather~numInstances * trainPercentage / 100,,0)
testSize  = weather~numInstances - trainSize
trainData = .Instances~new(weather, 0, trainSize)
testData  = .Instances~new(weather, trainSize, testSize)
-- Compute Decision Tree
lr = .bsf~new("weka.classifiers.functions.LinearRegression")
-- both, trainData and testData have the same structure (same attributes)
trainData~setClassIndex(trainData~numAttributes-1)
lr~buildClassifier(trainData)    -- build the model
say lr~toString                  -- print out the result
-- Evaluate Model
eval = .bsf~new("weka.classifiers.Evaluation",trainData)
testData~setClassIndex(testData~numAttributes-1)   -- set the dependant variable of the testing data (in this case the last column)
-- this version of evaluateModel expects three (!) arguments
noPredictions=bsf.createJavaArray("java.lang.Object",0)  -- create a 0 capacity Java array
eval~evaluateModel(lr, testData, noPredictions) -- evaluate the model
say "=== toSummaryString ===" eval~toSummaryString

::requires "BSF.CLS"  -- get Java-ooRexx bridge
```

Weka – Linear Regression

Program Output

```
NOTE: Picked up JDK_JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=jdk.compiler/com.sun.tools.javac.processing=ALL-UNNAMED
Jän. 23, 2026 12:53:25 AM com.github.fommil.netlib.ARPACK <clinit>
WARNING: Failed to load implementation from: com.github.fommil.netlib.NativeSystemARPACK
Jän. 23, 2026 12:53:25 AM com.github.fommil.jni.JniLoader liberalLoad
INFO: successfully loaded C:\Users\ADMINI~1\AppData\Local\Temp\jniloader7451495622485683368netlib-native_ref-win-x86_64.dll
Jän. 23, 2026 12:53:26 AM com.github.fommil.netlib.BLAS <clinit>
WARNING: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
Jän. 23, 2026 12:53:26 AM com.github.fommil.jni.JniLoader load
INFO: already loaded netlib-native_ref-win-x86_64.dll
Jän. 23, 2026 12:53:26 AM com.github.fommil.netlib.LAPACK <clinit>
WARNING: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
Jän. 23, 2026 12:53:26 AM com.github.fommil.jni.JniLoader load
INFO: already loaded netlib-native_ref-win-x86_64.dll

Linear Regression Model

Price =

   1138.9569 * SquareMeters +
  57701.7943 * Bathrooms +
  -1570.6529
=== toSummaryString ===
Correlation coefficient                  0.9765
Mean absolute error                  17253.4261
Root mean squared error              20315.711
Relative absolute error                 21.1987 %
Root relative squared error             22.9693 %
Total Number of Instances                9
```

Weka – Data Visualisation

- There is csv data (`house_prices.csv`) about square meters, bed rooms, bath rooms and the price like this

```
SquareMeters,Bedrooms,Bathrooms,Price
139,3,2,250000
167,4,3,350000
111,2,1,180000
186,3,2,400000
204,4,3,380000
158,3,2,320000
130,2,1,220000
149,3,2,260000
177,4,3,410000
…
```

- Create a plot from the data and let the user interact with it

Weka – Data Visualisation

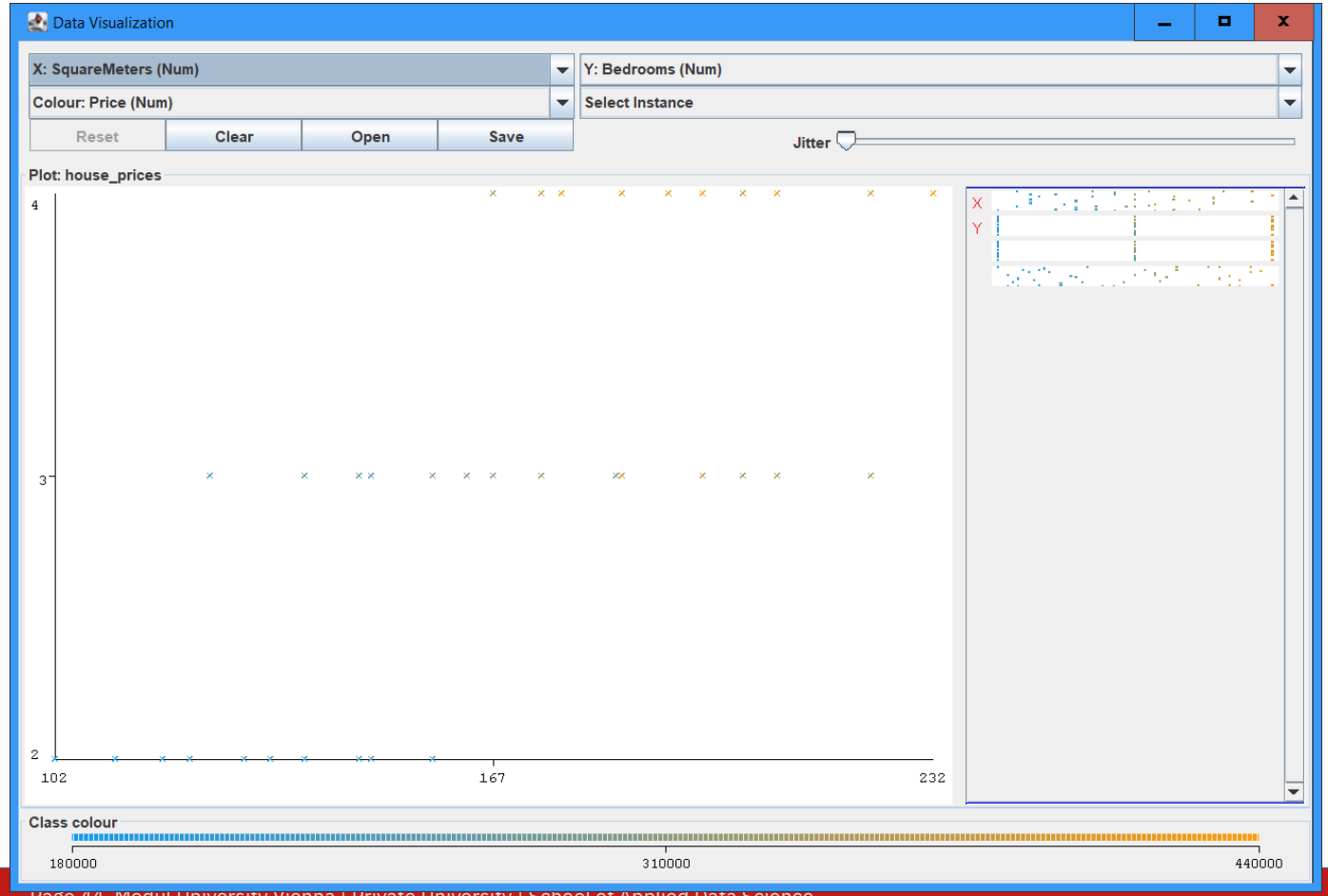
WU\02_PavicJacov\nutshells_examples\data_visualization\data_visualization.rxj

```
-- DataSource class is embedded in ConverterUtils
call bsf.import "weka.core.converters.ConverterUtils$DataSource", DataSource
call bsf.import "weka.core.Instances", Instances
call bsf.import "javax.swing.JFrame", JFrame

-- load packages
.bsf~new("weka.core.WekaPackageManager") ~loadPackages(.false) -- .false: quiet
-- import data set
housing = .DataSource~new("./house_prices.csv") ~getDataSet
-- create plotdata2D instance with the data as input
plotData = .bsf~new("weka.gui.visualize.PlotData2D", housing) ~~setPlotName("DATA")
-- create the visualization panel
panel = .bsf~new("weka.gui.visualize.VisualizePanel") ~~addPlot(plotData)
-- create a JFrame to hold the visualization panel
frame = .JFrame~new("Data Visualization")
frame~setDefaultCloseOperation(.JFrame~EXIT_ON_CLOSE)   -- make JFRAME closable
frame~setSize(1600, 1200)
frame~getContentPane~add(panel)     -- add panel to contentPane
frame~~setVisible(.true) ~~toFront  -- show
say "Hit enter to continue"
parse pull .                        -- otherwise ooRexx ends and Java with it

::requires "BSF.CLS"    -- get Java-ooRexx bridge
```


Prof. Rony G. Flatscher

Weka – Data Visualization

house_prices.csv



Weka – Applying Multiple Classifiers

- Applying multiple classifiers on weather data
  - Show default values and the synopsis of all available options per classifier
  - Show results for each classifier
  - Classifiers
    - J48 (decision tree)
    - IBk (k-nearest neighbor)
    - Logistic (logistic regression)
    - Naive Bayes
    - Random forest
    - SMO (support vector machine)
- Hint: to get at the JavaDocs search with the needles "javadoc weka", e.g.,

  javadoc weka Instances

Weka – Applying Multiple Classifiers

Weather Data

Weather.nominal.arff

```
@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no
~
~
```


Prof. Rony G. Flatscher

Weka – Applying Multiple Classifiers 1/2

WU\02_PavicJacov\nutshells_examples\supervised_learning\classifyAndEvaluateWeather.rxj

```
-- DataSource is an inner (embedded class) of the ConverterUtils class
call bsf.import "weka.core.converters.ConverterUtils$DataSource", DataSource
call bsf.import "weka.core.Instances", Instances

-- load packages
.bsf~new("weka.core.WekaPackageManager") ~loadPackages(.false) -- .false: quiet

-- import weather data set
weather = .DataSource~new("./weather.nominal.arff") ~getDataSet
say "=== weather data ==="
say weather~toString
say
say "there are" pp(weather~numInstances) "instances"
seed    = 0
weather~randomize( .bsf~new("java.util.Random",seed) )

trainPercentage = 60
trainSize = format(weather~numInstances * trainPercentage / 100,,0)
testSize  = weather~numInstances - trainSize
trainData = .Instances~new(weather, 0, trainSize)
testData  = .Instances~new(weather, trainSize, testSize)

… continued …
```

Weka – Applying Multiple Classifiers 2/2

WU\02_PavicJacov\nutshells_examples\supervised_learning\classifyAndEvaluateWeather.rxj

```
   -- array of classifiers to use    … continued …
arrClz="weka.classifiers.trees.J48",            -  -- decision tree
       "weka.classifiers.lazy.IBk",             -  -- k-nearest neighbor
       "weka.classifiers.functions.Logistic",   -  -- logistic regression
       "weka.classifiers.bayes.NaiveBayes",     -  -- naive Bayes
       "weka.classifiers.trees.RandomForest",   -  -- random forest
       "weka.classifiers.functions.SMO"            -- support vector machine
say "=== classifying and evaluation ==="
do counter c clz over arrClz     -- iterate and apply classifiers
   className=clz~substr(clz~lastPos('.')+1)
   say "--- classifier #" c":" pp(className) "("clz")" "---"
   model = .bsf~new(clz)   -- create instance, show options and synopsis
   say "using (default) options:" pp(.java.lang.String~join(" ",model~getOptions))
   do counter c2 option over model~listOptions
      say "   #" c2~right(2)":" pp(option~name)":" pp(option~synopsis)
   end

   testData ~setClassIndex(testData~numAttributes-1)  -- set the dependant variable
   trainData~setClassIndex(trainData~numAttributes-1)

      -- both, trainData and testData have the same structure (same attributes)
   model~buildClassifier(trainData) -- build the model
   say pp(model~toString)           -- print out the result
      -- Evaluate Model
   eval = .bsf~new("weka.classifiers.Evaluation",trainData)
      -- this version of evaluateModel expects three (!) arguments
   noPredictions=bsf.createJavaArray("java.lang.Object",0)  -- create an empty Java array
   eval~evaluateModel(model, testData, noPredictions)       -- evaluate the model
   say "___ toSummaryString ___" eval~toSummaryString
   say eval~toMatrixString          -- print out the confusion matrix
   say
end
::requires "BSF.CLS"  -- get Java-ooRexx bridge
```

Weka – Applying Multiple Classifiers

Program Output 1/8

```
NOTE: Picked up JDK_JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=jdk.compiler/com.sun.tools.javac.processing=ALL-UNNAMED
Jän. 25, 2026 8:35:54 PM com.github.fommil.netlib.ARPACK <clinit>
WARNING: Failed to load implementation from: com.github.fommil.netlib.NativeSystemARPACK
Jän. 25, 2026 8:35:55 PM com.github.fommil.jni.JniLoader liberalLoad
INFO: successfully loaded C:\Users\ADMINI~1\AppData\Local\Temp\jniloader1379002689571402209 4netlib-native_ref-win-x86_64.dll
=== weather data ===

@relation weather.symbolic

@attribute outlook {sunny,overcast,rainy}
@attribute temperature {hot,mild,cool}
@attribute humidity {high,normal}
@attribute windy {TRUE,FALSE}
@attribute play {yes,no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no

there are [14] instances

… continued …
```


Prof. Rony G. Flatscher

Weka – Applying Multiple Classifiers

Program Output 2/8

```
=== classifying and evaluation ===
--- classifier # 1: [J48] (weka.classifiers.trees.J48) ---
using (default) options: [-C 0.25 -M 2]
   #  1: [U]: [-U]
   #  2: [O]: [-O]
   #  3: [C]: [-C <pruning confidence>]
   #  4: [M]: [-M <minimum number of instances>]
   #  5: [R]: [-R]
   #  6: [N]: [-N <number of folds>]
   #  7: [B]: [-B]
   #  8: [S]: [-S]
   #  9: [L]: [-L]
   # 10: [A]: [-A]
   # 11: [J]: [-J]
   # 12: [Q]: [-Q <seed>]
   # 13: [-doNotMakeSplitPointActualValue]: [-doNotMakeSplitPointActualValue]
   # 14: [output-debug-info]: [-output-debug-info]
   # 15: [-do-not-check-capabilities]: [-do-not-check-capabilities]
   # 16: [num-decimal-places]: [-num-decimal-places]
   # 17: [batch-size]: [-batch-size]
[J48 pruned tree
------------------
: yes (8.0/2.0)

Number of Leaves  :     1

Size of the tree :      1

… continued …
```

```
___ toSummaryString ___
Correctly Classified Instances           3               50      %
Incorrectly Classified Instances         3               50      %
Kappa statistic                          0
Mean absolute error                      0.5
Root mean squared error                  0.559
Relative absolute error                100       %
Root relative squared error            103.8068 %
Total Number of Instances                6

=== Confusion Matrix ===

 a b   <-- classified as
 3 0 | a = yes
 3 0 | b = no

… continued …
```


Prof. Rony G. Flatscher

Weka – Applying Multiple Classifiers

Program Output 3/8

```
--- classifier # 2: [IBk] (weka.classifiers.lazy.IBk) ---

using (default) options: [-K 1 -W 0 -A
weka.core.neighboursearch.LinearNNSearch -A "weka.core.EuclideanDistance -R
first-last"]
   #  1: [I]: [-I]
   #  2: [F]: [-F]
   #  3: [K]: [-K <number of neighbors>]
   #  4: [E]: [-E]
   #  5: [W]: [-W <window size>]
   #  6: [X]: [-X]
   #  7: [A]: [-A]
   #  8: [output-debug-info]: [-output-debug-info]
   #  9: [-do-not-check-capabilities]: [-do-not-check-capabilities]
   # 10: [num-decimal-places]: [-num-decimal-places]
   # 11: [batch-size]: [-batch-size]
[IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
]
___ toSummaryString ___
Correctly Classified Instances           3               50      %
Incorrectly Classified Instances         3               50      %
Kappa statistic                          0
Mean absolute error                      0.5529
Root mean squared error                  0.6703
Relative absolute error                110.582  %
Root relative squared error            124.471  %
Total Number of Instances                6

=== Confusion Matrix ===

 a b   <-- classified as
 2 1 | a = yes
 2 1 | b = no
```

… continued on next page ….

Weka – Applying Multiple Classifiers

Program Output 4/8

```
--- classifier # 3: [Logistic] (weka.classifiers.functions.Logistic) ---
using (default) options: [-R 1.0E-8 -M -1 -num-decimal-places 4]
   #  1: [C]: [-C]
   #  2: [S]: [-S]
   #  3: [R]: [-R <ridge>]
   #  4: [M]: [-M <number>]
   #  5: [output-debug-info]: [-output-debug-info]
   #  6: [-do-not-check-capabilities]: [-do-not-check-capabilities]
   #  7: [num-decimal-places]: [-num-decimal-places]
   #  8: [batch-size]: [-batch-size]
[Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
                           Class
Variable                     yes
=================================
outlook=sunny            -21.3057
outlook=overcast           8.4056
outlook=rainy             14.5813
temperature=hot          -29.2883
temperature=mild          -5.264
temperature=cool          19.2828
humidity=normal           11.8799
windy=FALSE               -8.4056
Intercept                 20.758

Odds Ratios...
                           Class
Variable                     yes
=================================
outlook=sunny                  0
outlook=overcast        4471.9088
outlook=rainy        2150615.3441
temperature=hot                0
temperature=mild          0.0052
temperature=cool    236813942.4235
humidity=normal        144338.6099
windy=FALSE                 0.0002
]

___ toSummaryString ___
Correctly Classified Instances           3               50      %
Incorrectly Classified Instances         3               50      %
Kappa statistic                          0
Mean absolute error                      0.506
Root mean squared error                  0.7072
Relative absolute error                101.2007 %
Root relative squared error            131.3231 %
Total Number of Instances                6

=== Confusion Matrix ===

 a b   <-- classified as
 2 1 | a = yes
 2 1 | b = no

… continued on next page …
```

Weka – Applying Multiple Classifiers

Program Output 5/8

```
--- classifier # 4: [NaiveBayes] (weka.classifiers.bayes.NaiveBayes) ---
using (default) options: []
   #  1: [K]: [-K]
   #  2: [D]: [-D]
   #  3: [O]: [-O]
   #  4: [output-debug-info]: [-output-debug-info]
   #  5: [-do-not-check-capabilities]: [-do-not-check-capabilities]
   #  6: [num-decimal-places]: [-num-decimal-places]
   #  7: [batch-size]: [-batch-size]
[Naive Bayes Classifier

               Class
Attribute        yes    no
               (0.7) (0.3)
============================
outlook
  sunny           2.0   3.0
  overcast        3.0   1.0
  rainy           4.0   1.0
  [total]         9.0   5.0

temperature
  hot             1.0   2.0
  mild            4.0   2.0
  cool            4.0   1.0
  [total]         9.0   5.0

humidity
  high            3.0   3.0
  normal          5.0   1.0
  [total]         8.0   4.0

temperature=cool    236813942.4235
humidity=normal        144338.6099windy
  TRUE            3.0   1.0
  FALSE           5.0   3.0
  [total]         8.0   4.0

]

___ toSummaryString ___
Correctly Classified Instances           3               50      %
Incorrectly Classified Instances         3               50      %
Kappa statistic                          0
Mean absolute error                      0.5261
Root mean squared error                  0.6131
Relative absolute error                105.2207 %
Root relative squared error            113.8522 %
Total Number of Instances                6


=== Confusion Matrix ===

 a b   <-- classified as
 2 1 | a = yes
 2 1 | b = no

… continued on next page …
```


Prof. Rony G. Flatscher

Weka – Applying Multiple Classifiers

Program Output 6/8

```
--- classifier # 5: [RandomForest] (weka.classifiers.trees.RandomForest) ---
using (default) options: [-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1]
   #  1: [P]: [-P]
   #  2: [O]: [-O]
   #  3: [store-out-of-bag-predictions]: [-store-out-of-bag-predictions]
   #  4: [output-out-of-bag-complexity-statistics]: [-output-out-of-bag-complexity-
statistics]
   #  5: [print]: [-print]
   #  6: [attribute-importance]: [-attribute-importance]
   #  7: [I]: [-I <num>]
   #  8: [num-slots]: [-num-slots <num>]
   #  9: [K]: [-K <number of attributes>]
   # 10: [M]: [-M <minimum number of instances>]
   # 11: [V]: [-V <minimum variance for split>]
   # 12: [S]: [-S <num>]
   # 13: [depth]: [-depth <num>]
   # 14: [N]: [-N <num>]
   # 15: [U]: [-U]
   # 16: [B]: [-B]
   # 17: [output-debug-info]: [-output-debug-info]
   # 18: [-do-not-check-capabilities]: [-do-not-check-capabilities]
   # 19: [num-decimal-places]: [-num-decimal-places]
   # 20: [batch-size]: [-batch-size]
[RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-
capabilities]

___ toSummaryString ___
Correctly Classified Instances           3               50      %
Incorrectly Classified Instances         3               50      %
Kappa statistic                          0
```

```
Mean absolute error                      0.553
Root mean squared error                  0.6447
Relative absolute error                110.6083 %
Root relative squared error            119.722  %
Total Number of Instances                6

=== Confusion Matrix ===

 a b   <-- classified as
 2 1 | a = yes
 2 1 | b = no
```

… continued on next page …

Weka – Applying Multiple Classifiers

Program Output 7/8

```
--- classifier # 6: [SMO] (weka.classifiers.functions.SMO) ---
using (default) options: [-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K
weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007 -calibrator
weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4]
   #  1: [no-checks]: [-no-checks]
   #  2: [C]: [-C <double>]
   #  3: [N]: [-N]
   #  4: [L]: [-L <double>]
   #  5: [P]: [-P <double>]
   #  6: [M]: [-M]
   #  7: [V]: [-V <double>]
   #  8: [W]: [-W <double>]
   #  9: [K]: [-K <classname and parameters>]
   # 10: [calibrator]: [-calibrator <scheme specification>]
   # 11: [output-debug-info]: [-output-debug-info]
   # 12: [-do-not-check-capabilities]: [-do-not-check-capabilities]
   # 13: [num-decimal-places]: [-num-decimal-places]
   # 14: [batch-size]: [-batch-size]
   # 15: []: [
Options specific to kernel weka.classifiers.functions.supportVector.PolyKernel:]
   # 16: [E]: [-E <num>]
   # 17: [L]: [-L]
   # 18: [C]: [-C <num>]
   # 19: [output-debug-info]: [-output-debug-info]
   # 20: []: [
Options specific to calibrator weka.classifiers.functions.Logistic:]
   # 21: [C]: [-C]
   # 22: [S]: [-S]
   # 23: [R]: [-R <ridge>]
   # 24: [M]: [-M <number>]
   # 25: [output-debug-info]: [-output-debug-info]
   # 26: [-do-not-check-capabilities]: [-do-not-check-capabilities]
   # 27: [num-decimal-places]: [-num-decimal-places]
   # 28: [batch-size]: [-batch-size]
[SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: yes, no

BinarySMO

Machine linear: showing attribute weights, not support vectors.

         0.8998 * (normalized) outlook=sunny
 +      -0.2996 * (normalized) outlook=overcast
 +      -0.6002 * (normalized) outlook=rainy
 +       0.6002 * (normalized) temperature=hot
 +       0.1002 * (normalized) temperature=mild
 +      -0.7004 * (normalized) temperature=cool
 +      -0.7004 * (normalized) humidity=normal
 +       0.2996 * (normalized) windy=FALSE
 -       0.7996

Number of kernel evaluations: 28 (81.579% cached)

… continued on next page …
```

Weka – Applying Multiple Classifiers

Program Output 8/8

```
___ toSummaryString ___
Correctly Classified Instances           4               66.6667 %
Incorrectly Classified Instances         2               33.3333 %
Kappa statistic                          0.3333
Mean absolute error                      0.3333
Root mean squared error                  0.5774
Relative absolute error                 66.6667 %
Root relative squared error            107.2113 %
Total Number of Instances                6


=== Confusion Matrix ===

 a b   <-- classified as
 3 0 | a = yes
 2 1 | b = no
```


Prof. Rony G. Flatscher

Roundup

- Machine learning (ML) with Weka
  - GUI
  - Command line interface (CLI), scripts
  - Huge Java class library related to all aspects of ML
  - Weka can be used on Windows, Apple, Linux, and everywhere where Java exists
  - Meant to be used with Java, however can be used by scripting languages as well
    - The Java-ooRexx bridge BSF4ooRexx allows to use the Weka class library and its numerouse packages
    - Weka sample code demonstrates the use of scripting languages like Groovy or Jython (a Java implementation of Python)
  - A great tool to experiment, research, and teach machine learning ...

    ... and it is free and open source!