Tutorial on Machine Learning.

Part 1. Impact of dataset composition on models performance

G. Marcou, N. Weill, D. Horvath, D. Rognan, A. Varnek

1 Introduction

Predictive performance of QSAR model depends not only on the way how this model has been built and validated (descriptors, machine-learning method, variable selection procedure, scrambling, cross-validation, etc.) but also on similarity between the compounds of training and test sets. Indeed, a training set containing a limited number of compounds is never representative with respect to infinite chemical space. As a consequence, the learning of a chemical or biological property is often biased, *i.e.* reflect the training set composition. In order to assess predictive performance of the model, the bias between training and test sets could be analyzed using applicability domain or any other related approach.

In this tutorial, classifications models build on a training set for antibiotic activities are applied to one related and one unrelated external test sets. Difference between predictive performances of the models applied on related and unrelated sets is interpreted comparing descriptors distribution in all three sets (data profiling). All calculations will be performed using open source software Weka (data mining) [1] and R (graphical representation of the data) [2].
Description of data preprocessing (with ChemAxon) and descriptors generation (with OpenEye, MOE and ISIDA) is given in APPENDIXES 1 and 2. Some explanations on Weka graphical interface are given in APPENDIX 3.

1.1 Software

1.1.1 R

R is an open source environment for statistical computing and graphics [2].To run R, on Linux type `R`; on Windows click on the R icon. In order to facilitate use of R, it is recommended to copy the tutorial files the directory `C:\WORKDIR`. If the application is not run from this working directory, in the R command line area, enter the command:

```
setwd('<name of your working directory>')
```

1.1.2 Weka

The Weka software is a graphical user interface (GUI) to a Java library dedicated to data mining [1]. A large part of the data mining algorithm is accessible through this interface. It is recommended to use the latest developer version (3.5.7). The library is developed by the Waikato university (New Zealand). The Weka is a native bird of New Zealand, in danger of extinction.

Run Weka. Since Weka default memory allocation is insufficient to work with these tutorial datasets, some setting parameters should be modified.

Linux:

Open a terminal and type:

```
java -Xmx512m -jar weka.jar
```

There might be incompatibilities between the java runtime environment (JRE) on some Linux distributions. To avoid this, it is recommended to create a file in the user's home directory called `LookAndFeel.props`, containing the line:

```
Theme=javax.swing.plaf.metal.MetalLookAndFeel
```

Windows:

Modify the file `RunWeka` in the installation directory of Weka (root privilege required). Change the line:

```
maxheap=128m
```

by:

```
maxheap=512m
```

Then launch Weka by clicking on the window's **Start** button then in **Program/Weka 3.5.7**, click on **Weka 3.5**.

1.2 Datasets

Two datasets are used in this tutorial. The first one, stored in the directory `Cherkasov`, was published in reference [5]. It contains a set of antibacterials (521 cmpds) and decoys (2158 cmpds). The latter were selected from drugs of different therapeutic classes and for which no antibacterial activity is known. This set will be further split into a training set and a related test set.

The second is the external unrelated test set, stored in the directory `CN`. These data were extracted from the experimental screening of the Chimiothèque Nationale [4] for the antibacterial activity on *S. Aureus*.

2 Model building

2.1 Descriptors

Three sets of desriptors are used:

- trivial descriptors: 8 descriptors inspired by Shoishet et al.[5]: (the molecular weight, the number of heteroatoms, the number of cycles, the number of rotatable bonds, the number of rigid bonds, the number of hydrogen bonds acceptors, the number of hydrogen bonds donors, the 2D polar surface area);
- MOE 2D descriptors (optionally);
- ISIDA / SMF descriptors [8]: paths of 2 to 5 atoms in the molecular graph represented by

the element type (optionally).

2.2 Data files

Here, we provide some information about files both stored in the `Cherkasov` and CN directories and which are supposed to be created in course of the tutorial.

The `Cherkasov` directory contains the following files:

- `Cherkasov.sdf`: the same file in SDF format. The activity flag is stored in the SDF field: > <*Antibio*>;
- `Chekasov_std.sdf`: the same file in SDF format, standardized with Chemaxon's tools.
- `config.xml`: a configuration file for Chemaxon's standardizer [6];
- `PassAll.txt`: a configuration file for OpenEye's Filter [7];
- `CherkasovDB.mdb`: the MOE database for this dataset;
- `Cherkasov_MOEDsc.csv`: the 2D descriptors calculated by MOE;
- `Cherkasov_SMF.arff`: the ISIDA/SMF fragment descriptors;
- `Cherkasov_SMF.hdr`: file identifying each ISIDA/SMF [8] fragment index to a string description of the fragment;
- `Cherkasov_MOEDsc_80p.arff`: MOE descriptors of the training set, a subset of 80% of the original dataset;
- `Cherkasov_MOEDsc_20p.arff`: MOE descriptors of the related test set, the remaining 20% of the original dataset;
- `Cherkasov_trivial_80p.arff`: simple descriptors of the training set, a subset of 80% of the original dataset;
- `Cherkasov_trivial_20p.arff`: simple descriptors of the related test set, the remaining 20% of the original dataset;
- `trivial.tbl`: output properties table of OpenEye's filter manipulations;
- `trivial.csv`: the same cleaned data to keep only the column used in this tutorial.
- `trivial.arff`: the same processed data in ARFF fromat.

Content of the directory `CN`:

- `CN.sdf`: raw dataset file containing the flag for antibacterial activity (0 not active, 1 active) in an SDF field (> <*Antibio*>);
- `CN_std.sdf`: standardised file;
- `CN_trivial.arff`: trivial descriptors file (ARFF format)
- `CN_MOEDsc.arff`: MOE 2D descriptors file (ARFF format)
- `CN_SMF.arff`: ISIDA/SMF descriptors file in ARFF format.

• `CN_SMF.hdr`: file identifying each ISIDA/SMF fragment descriptors index to a string describing it.

2.3 Obtaining and validation of the models

Typically, in QSAR studies an initial parent dataset is split into training and test sets. The models are build on the training set followed by their validation on the test set.

2.3.1 Extracting the training and theexternal related test set from the Cherkasov dataset.

Instructions

1. Open Weka;
2. In the tool bar, click on **Application** then on **Explorer**;

Transformation "Numeric To Nominal";

3. in the **Explorer** window, click on **Open file** and select `Cherkasov /trivial.csv`;
4. select the attribute *name* then click on **Remove**;
5. click on **Choose**, then **Filter/unsupervised/attribute/NumericToNominal**;
6. click on the text zone set to **NumericToNominal**;
7. in the pop-up window, set the parameter **attributeIndices** to *last* then click on **OK**;
8. click on **Apply**;

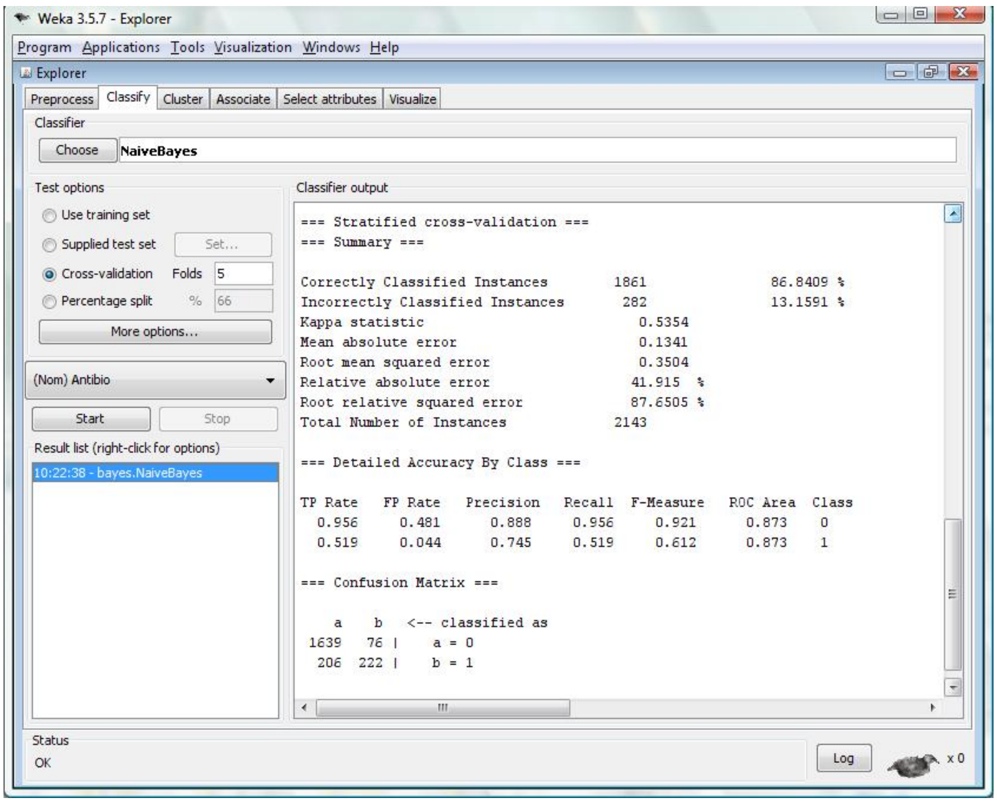
Split of the parent set into training and test sets

9. click on **Save** and name the new dataset as `trivial.arff`;
10. click on **Choose**;
11. select the **unsupervised**, **instance**, then **RemoveFolds** method;
12. click on the text zone set to **RemoveFolds**;
13. set **fold** to 1, **numFolds** to 5 and **seed** to 1;
14. click on **OK**;
15. click on **Apply**;
16. click on **Save** and name the new dataset as `Cherkasov_trivial_20p.arff`;
17. click on the text zone set to **RemoveFolds**;
18. set **inverSelection** to *True*;
19. click on **OK**;
20. click on **Apply**;
21. click on **Save** and name the new dataset as `Cherkasov_trivial_80p.arff`;

It is important to keep the same integer value (different from 0) for the seed of the random number generator and to keep it different from 0. This will unsure that the fold separation is always the same. If the seed were 0, the parent dataset would not be shuffled and therefore, resulting training and test sets would not be representative.

2.3.2 Build and analyze a Bayesian model

1. in the **Explorer** window, click on **Open file** and select `Cherkasov_trivial_80p.arff`;
2. select the tab **Classify**;
3. click on **Choose**, **classifiers** / **Bayes** / **NaiveBayes**;
4. click on the **Cross-validation** radio button and set the parameter to 5 folds;
5. click on **Start**
6. *see results on the screenshot below*


Classifier: NaiveBayes

Test options: Cross-validation, Folds 5

(Nom) Antibio

Result list: 10:22:38 - bayes.NaiveBayes

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1861               86.8409 %
Incorrectly Classified Instances       282               13.1591 %
Kappa statistic                          0.5354
Mean absolute error                      0.1341
Root mean squared error                  0.3504
Relative absolute error                 41.915  %
Root relative squared error             87.6505 %
Total Number of Instances             2143

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.956     0.481     0.888      0.956     0.921      0.873    0
  0.519     0.044     0.745      0.519     0.612      0.873    1

=== Confusion Matrix ===

    a    b   <-- classified as
 1639   76 |    a = 0
  206  222 |    b = 1
```

Status: OK

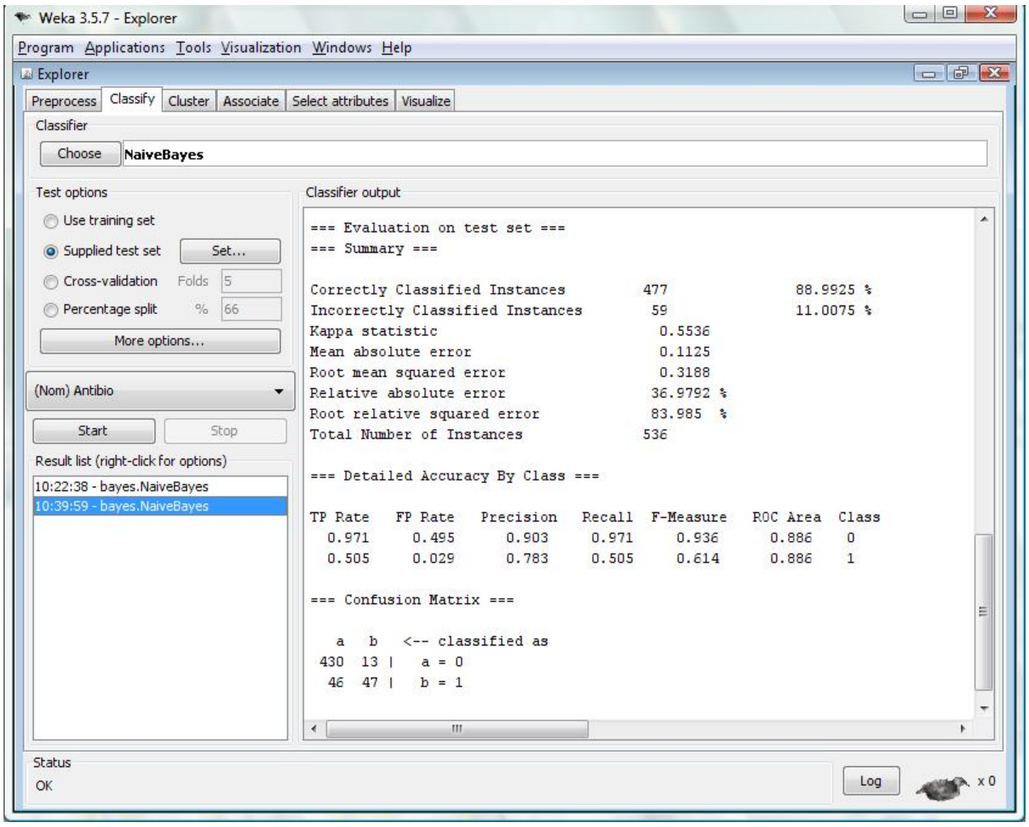
Results can be analyzed by right clicking on the corresponding line in the **Result list** window. In the contextual menu that opens, one can access to some useful options:

- Visualize Threshold Curve (Receiver Operator Curves)
- Visualize Cost Curve
- Visualize Margin Curve
- Save Model
- Load Model

Re-evaluate

External validation on the related test set

1. select the tab **Classify**;
2. click on **Choose**, **classifiers** / **Bayes** / **NaiveBayes**;
3. click on the button **Set** and select the file `Cherkasov_trivial_20p.arff`;
4. click on **Start**
5. *see results on the screenshot below*



Classifier: NaiveBayes

Test options: Supplied test set; Cross-validation Folds 5; Percentage split % 66

(Nom) Antibio

Result list:
10:22:38 - bayes.NaiveBayes
10:39:59 - bayes.NaiveBayes

Status: OK

```
=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances         477               88.9925 %
Incorrectly Classified Instances        59               11.0075 %
Kappa statistic                          0.5536
Mean absolute error                      0.1125
Root mean squared error                  0.3188
Relative absolute error                 36.9792 %
Root relative squared error             83.985  %
Total Number of Instances              536

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.971     0.495      0.903     0.971     0.936      0.886    0
  0.505     0.029      0.783     0.505     0.614      0.886    1

=== Confusion Matrix ===

   a   b   <-- classified as
 430  13 |   a = 0
  46  47 |   b = 1
```

2.4 External validations on unrelated test set

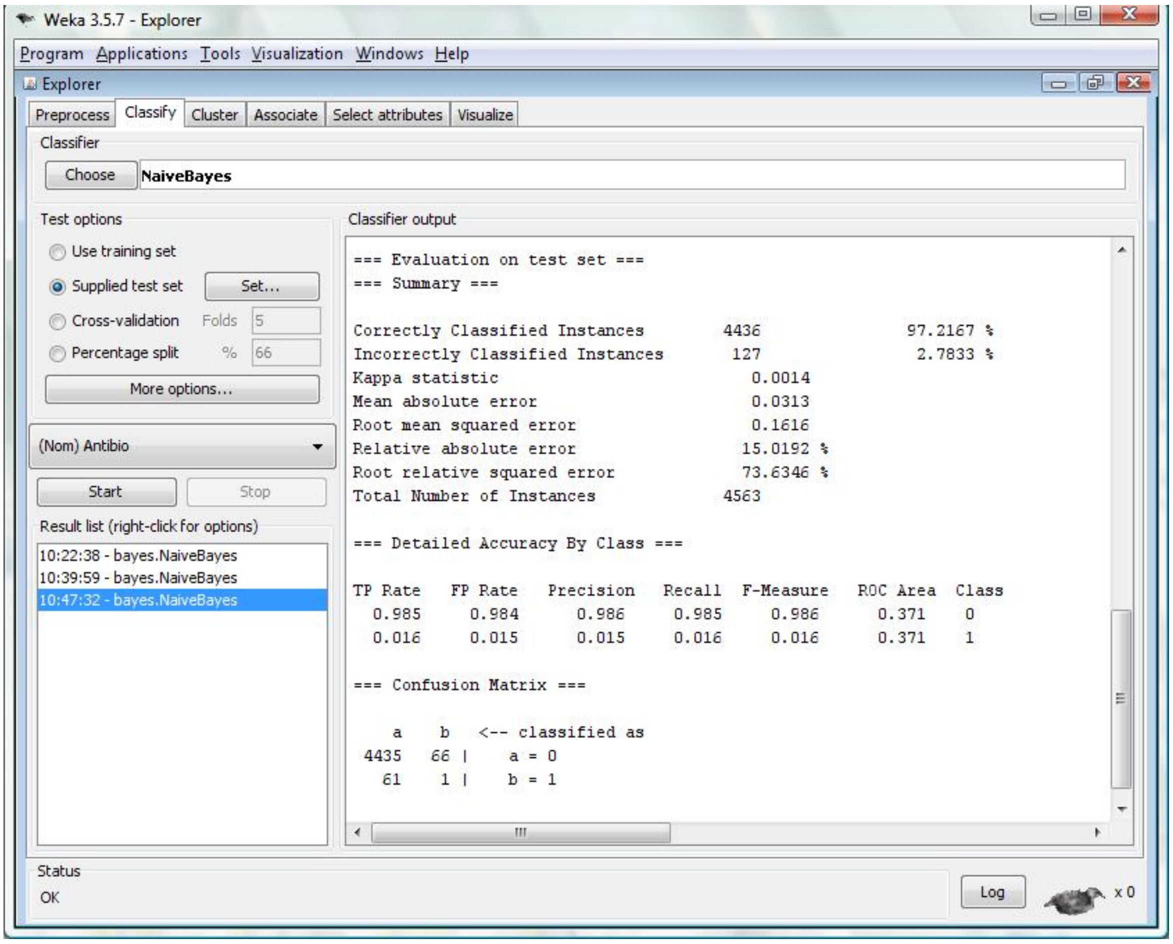
The CN dataset, which is highly diverse and completely unrelated to the previous ones, is used here as support for a real virtual screening exercise, challenging the model to predict the antibiotics therein. Since this set never served to influence the model buildup in any way, it is irrelevant that experimental screening actually preceded virtual screening in this example.

Save the model

1. righ click on the last line of the **Result list;**
2. in the contextual menu, choose **Save model** and save the model as `trivial.model`.

Apply the model

1. click on **Load model**;
2. select the file `trivial.model`;
3. click on the button **Set** and select the file `CN/trivial.arff`;
4. righ click on the last line of the **Result list**;
5. in the contextual menu, choose **Re-evaluate model on current test set**.
6. *see results on the screenshot below*



NaiveBayes

Supplied test set (selected); Folds 5; % 66

(Nom) Antibio

10:22:38 - bayes.NaiveBayes
10:39:59 - bayes.NaiveBayes
10:47:32 - bayes.NaiveBayes

```
=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances        4436               97.2167 %
Incorrectly Classified Instances       127                2.7833 %
Kappa statistic                          0.0014
Mean absolute error                      0.0313
Root mean squared error                  0.1616
Relative absolute error                 15.0192 %
Root relative squared error             73.6346 %
Total Number of Instances             4563

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.985     0.984      0.986     0.985     0.986      0.371    0
  0.016     0.015      0.015     0.016     0.016      0.371    1

=== Confusion Matrix ===

    a    b   <-- classified as
 4435   66 |    a = 0
   61    1 |    b = 1
```

Status
OK

Exercises (optional)

1. Repeat the same steps with ISIDA fragmental descriptors or MOE descriptors;
2. repeat these steps using the decision tree **J4.8**;
3. repeat these steps using the decision tree **J4.8** and the ISIDA descriptors or MOE descriptors.

It is remarkable that more sophisticated descriptors and data mining methods will not change the situation.

3 Detection of the bias: profiling of data sets.

The file `trivial.arff` in the `Cherkasov` directory will be examined. The file will be imported into Weka:

1. activate the **preproces** tab in Weka;
2. click on **Open file** and choose the file `trivial.csv`.

The distributions can be displayed as histograms (tab **preprocess** the button **visualise all**) or displayed as correlograms (tab **visualise**).

The graphical capacities of Weka are very limited. Other softwares are better suited for graphical display of dataset properties. Here we use R.

To start, use the command to set the working directory to the root directory of the tutorial. For instance, Windows users that followed the recommendations in the introduction shall type:

```
setwd('C:/WORKDIR/TutorialMarcouWeill/')
```

Then execute the R `RBoxplot.R` script:

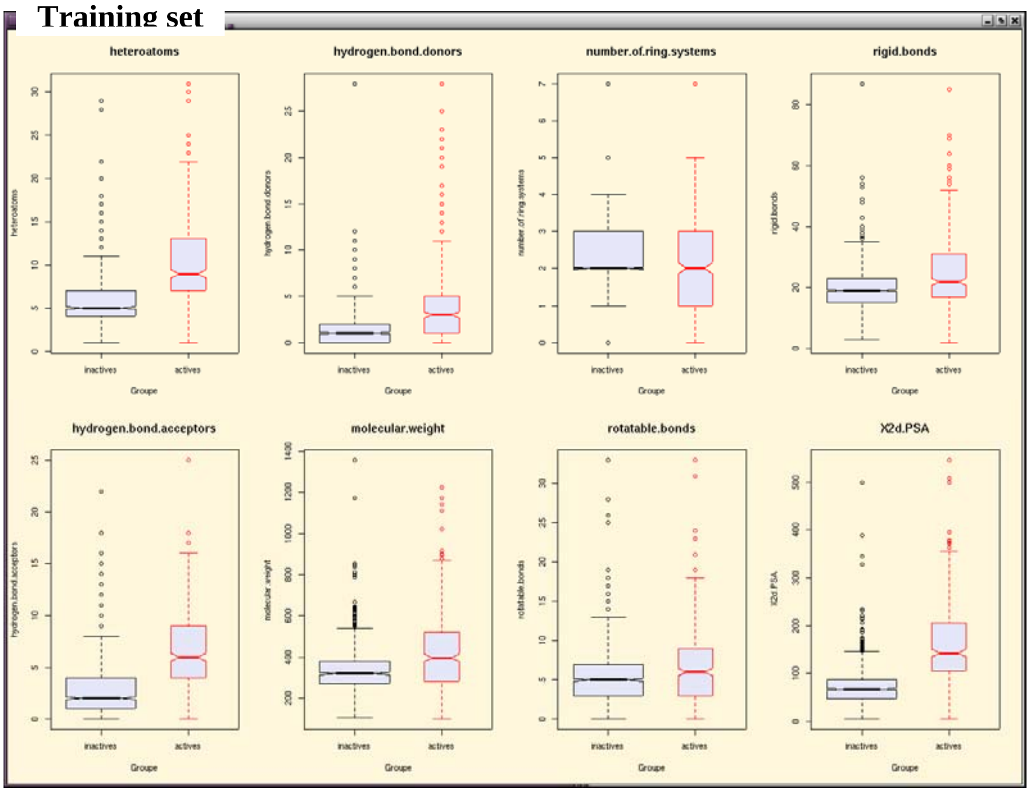
```
source('RBoxplot.R')
```

This will create a function to plot box plots of the data. Use the function to plot the profile of the training, related and unrelated test sets respectively:

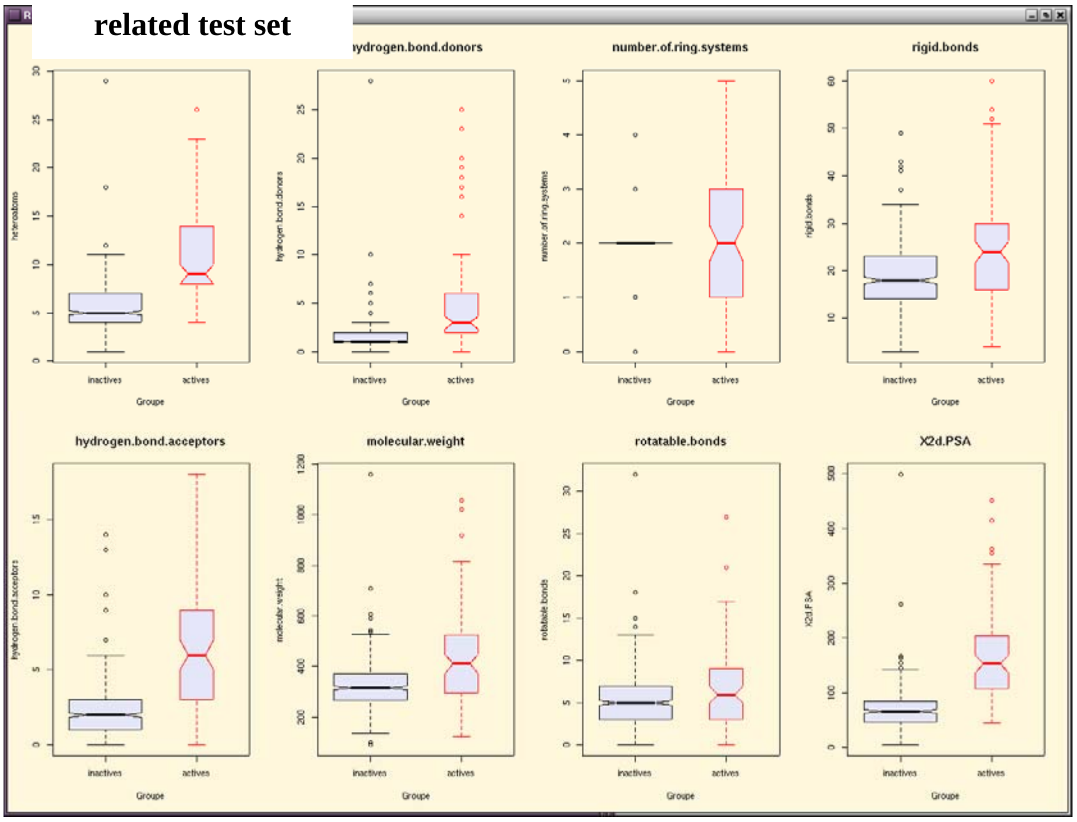
```
plotprofile(``Cherkasov/trivial_80p.csv'',''Inactive'',''Active'
')
X11()
plotprofile(``Cherkasov/trivial_20p.csv'',''Inactive'',''Active'
')
X11()
plotprofile(``CN/trivial.csv'',''Inactive'',''Active'')
```

On the screenshots below one can easily find similarity of descriptors profiles for training and related test sets, whereas that for unrelated test set looks rather different. It should be noted that in related sets (belonging to Cherkasov's data [3]]) descriptors well separate actives and inactives, whereas in unrelated test set (Chimiothèque Nationale [4]) no distinct separation is observed.

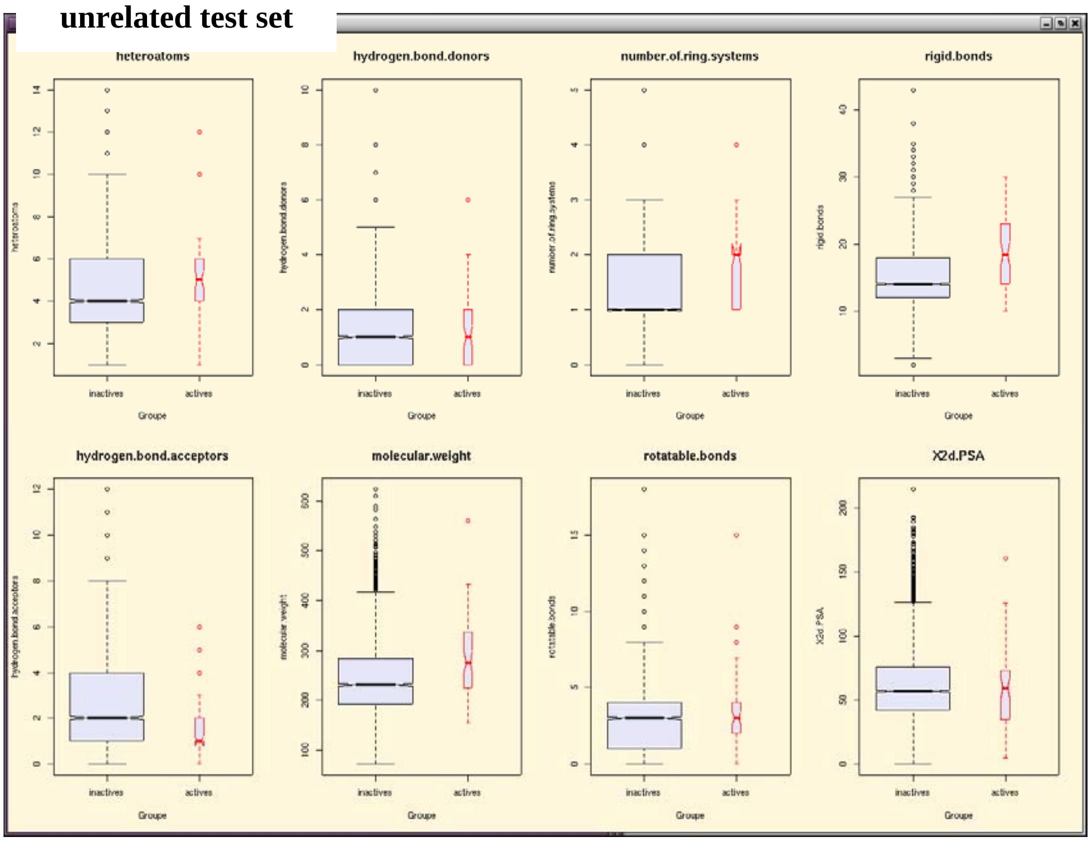
Training set



related test set



unrelated test set



4 Conclusion

Descriptors profiling allowed us to interpret predictive performance of the models with respect to the bias between training and test set. However, this solution is too coarse and doesn't characterize sufficiently the datasets. Applicability domains approaches are more relevant tools to treat the problem of datasets bias.

Specifically, for classification problems, the two levels of bias should be outlined

- the representation of limited subsets of actives. This yields the false impression that *only* the visited chemical space zones may harbor actives, whereas radically different chemotypes are nevertheless carriers of an activity, but cannot be learned on hand of the training set
- underrepresented "decoys", *i.e.* inactives in the immediate neighborhood of actives. This lets the user of the biased model believe that any structure in the neighborhood of an active is supposed to be an active, although in reality many small structural changes applied to an active may cause an activity loss. Even if *most* of small structural changes do not cause an activity loss (so that the similarity principle holds), the number of those who do, may still lead to an enormous set of near-identical inactives.

5. Further readings

A special issue of the Journal of Computer-Aided Molecular Design is dedicated to bias in drug design: J. Comp.-Aid. Mol. Design, Volume 22, Number 3-4, March 2008

References

[1] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, 2005.

[2] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[3] A. Cherkasov. Can 'bacterial-metabolite-likeness' model improve odds of 'in silico' antibiotic discovery? *J. Chem. Inf. Model.*, 46:1214–1222, 2006.

[4] *Chimiothèque Nationale*, Chimiotheque-nationale.enscm.fr, 2007.

[5] Huang N., Shoichet B. K., Irwin J. J., Benchmarking Sets for Molecular Docking. *Journal of Medicinal Chemistry*, 49(23):6789-6801, 2006.

[6] ChemAxon. Standardize. http://www.chemaxon.com, 2007, version 3.2.10.

[7] OpenEye Scientific Software Inc., Santa Fe, MA, USA. Filter. www.eyesopen.com. version 2.0.

[8] ISIDA project. http://infochim.u-strasbg.fr/recherche/isida/index.php

Appendix 1. Data preprocessing

Software used

- OpenEye's Filter: a software for fast chemical library filtering and analysis;
- Chemaxon's Standardizer: a software for custom chemical library cleaning;
- CCG's MOE: a general purpose bio/chemoinformatics environment;

OpenEye and Chemaxon offer licenses for free to academics, under reasonable conditions. CCG offers MOE licenses for a very affordable price to academics.

The use of commercial products is for the purpose of example. Nothing presented here is an exclusive feature of these programs and similar manipulations can be realized using concurrent products.

Data Standardization

It is not difficult to define some general rules for data standardization. Nonetheless, particular attention should be paid to:

- correctness of Lewis representations;
- aromaticity;
- presence of multiple chemical species;
- salts and counter-ions;
- tautomers;
- protonation;
- standardizations of key chemical functions;
- stereochemistry;
- conformers;

ChemAxon Standardizer is used to clean 2D structures.

Standardize

1. Open Standardizer;
2. load the file `Cherkasov.sdf` (**AddFile**), click on **Next**;
3. click on **Open Configuration**, load the file `Config.xml`;
4. take a look at the standardization rules (Click on a rule, then on **MarvinSketch** to edit the rule);
5. click on **Next**;
6. select an output file: `Cherkasov_std.sdf` -save it in SDF format-;
7. click on **Next** then on **Finish**;
8. consider the results.

Appendix 2. Generation of molecular descriptors

MOE and ISIDA are used to generate molecular descriptors. In this tutorial, only 2D descriptors are used.

MOE

1. Open the file `Cherkasov_std.sdf` (**File**, **Open**, **Import SD**);
2. in the **import** window, name the database as `CherkasovDB.mdb` then click on **scan** then on **OK**;
3. in the **Database Viewer**, click on **Compute** then on **Descriptors**;
4. in the **QuaSAR-Descriptor** window, click on **2D** and select all (*Shift+Click*) then **OK**;
5. in **Database Viewer**, right click on the header *"Antibio"* then, in the contextual menu, on **Move Field** and click on the name of the last column;
6. in the main menu, click on **File**, then on **Export**.
7. in the **Output format** menu, select **delimited ASCII database** then, as delimiter, select **Comma** and name the output file as `Cherkasov_MOEDsc.csv`.

MOE possess a QSAR module allowing to construct immediately a model. But in the following we work with Weka.

ISIDA/Fragmentor

On a command line type:

```
Fragmentor -i Cherkasov_std.sdf -o Cherkasov_SMF -t 1 -l 2 -u 5
-s Antibio -f SVM
```

This will generate an SVM file containing for each molecule a line containing its descriptors values. The first column of an SVM file is the value of the property to model. The header file, containing a string describing each descriptor possesses the extension HDR.

Simple descriptors

Most chemoinformatics tools allow monitoring the distribution of "simple" descriptors. Simple descriptors must depend only of 2D representation of molecules, have a straightforward interpretation and not depend of the details of the molecule -for instance, they might not be able to distinguish structural isomers. This set of descriptors is often used to define a *profile* of chemical libraryies.

Here are some proposals of simple descriptors:

- the molecular weight;
- the number of heteroatoms;
- the number of cycles;
- the number of rotatable bonds;
- the number of rigid bonds;
- the number of hydrogen bonds acceptors;
- the number of hydrogen bonds donors;
- the 2D polar surface area.

This list is non-exhaustive and is proposed for the tutorial. As stated before it was inspired by the Dataset of Usefull Decoy methodology [5]. A model based on these descriptors can yet be of some interest as a filter or pre-screen of a very large chemical database

.

“Simple” descriptors can be generated with **OpenEye Filter.** This software works as a command line for Linux or a command line in DOS terminal for Windows (type *cmd* in the **Run** text area in the Windows **Start** menu). The use of a cygwin terminal for Windows is recommended.

The command line:

```
filter -in Cherkasov_std.sdf -out Cherkasov_filter.sdf -filter
filter_trivial.txt -table trivial.tbl
```

A file containing the count of all the above mentioned descriptors (and more) is created as `trivial.tbl`. The columns are tab separated. A version of this file restricted to the sole descriptors above mentioned is provided as `trivial.csv`. The last column of this file contains the antibacterial activity flag.

It is also possible to calculate these descriptors with **MOE**.

Appendix 3. Weka at first glance

The Weka interface consists first in a main menu bar. These menu concern general manipulations of the interface: close the program, manipulate native data files (ARFF format), visualize graphical results, and provide help... The only menu of interest for us is labeled `Application`. Through this menu, it is possible to access to 4 separate modes to treat datasets:

- « Simple CLI» : command line interface. The syntax is almost equivalent to Java code.
- « Explorer» : Visualization, treatment of datasets.
- « Experimenter» : Apply a variety of data mining approach to a variety of datasets.
- « Knowledge Flow » : Graphical elaboration of data pipelining. The concept is similar to Pipeline Pilot.

7.1 Simple CLI

The syntax of a command line is close to the Java programming language. This simplifies greatly the development of unitary tests in the course of development of project making use of Java classes of Weka. This tutorial will not provide introduction to Java development nor Weka scripting.

7.2 Knowledge Flow

This is a functional graphical representation of a dataset treatment. Datasets, data mining methods, display and results are the nodes of a graph. The edges represent paths that can be followed by the data. This is similar in spirit to Pipeline Pilot and is considered rather as an "alternative" to the Explorer module (see later).

7.3 Experimenter

The goal of this module is to provide a way to process many combinations of data mining approaches on many datasets. Computations are distributed on a grid if available. Datasets and results can be stored and accessed relational database such as MySQL. Methods performances are stored and can be compared on a statistical basis. This module is not meant to build models to be published. It is rather used to identify the most fruitful approaches to a data mining problem.

7.4 Explorer

The explorer module will be the one used in this tutorial. It allows accessing to almost all the methods in the software interactively. It can build models and validate them. Visualizations and analysis tools are available.