



Prepared under the PROSIL project by: Thomas Ferrari, Dario Cattaneo, Alessio Mauro Franchi and Giuseppina Gini

SARpy - User's Manual

October 26, 2015



SARpy - User's manual

© COPYRIGHT 2015 by Thomas Ferrari, Dario Cattaneo, Alessio Mauro Franchi and Giuseppina Gini as part of the PROSIL project.

If you publish a model created by SARpy please quote one of the following: [1], [2], [3].

ALL RIGHTS RESERVED

Contents

Introduction	i
1 Prerequisites	1
2 Installing and starting SARpy	3
2.1 W-SARpy	3
2.2 S-SARpy	4
3 Working with datasets	5
3.1 Loading a dataset	5
3.2 Managing the current dataset	7
4 Working with rulesets	11
4.1 S-SARpy	14
5 Predicting and validating	17
5.1 W-SARpy	17
5.2 S-SARpy	20
6 W-SARpy step-by-step	23
6.1 Preparing the CSV dataset file	23
6.2 W-SARpy	25
6.3 S-SARpy	38
Bibliography	43

Introduction

Welcome to the SARpy user guide; this brief manual will introduce you to the SARpy tool, providing you with the basic knowledge for using this software. An illustrated step-by-step example is also provided in the last section.

SAR (Structure-Activity Relationships) models typically make use of rules, created by experts, to check for the presence of some specific structural fragments, called Structural Alerts (SA), already known to be responsible for the property under investigation.

SARpy (SAR in python) is a new ad hoc approach to automatically generate SAR models by finding the relevant rules from data, without any a priori knowledge. The algorithm generates substructures of arbitrary complexity, and automatically selects the fragments to become SAs on the basis of their prediction performance on a training set. Making a model requires to give SARpy a training set of molecular structures expressed in the SMILES notation, with their experimental activity binary labels.

Automatically SARpy generates rules in three steps:

- Fragmentation: this recursive algorithm computes every substructure in the molecules of the input set; min and max length of the substructures are user defined;
- Evaluation: substructures are ordered according to their potential role as SA;
- Rule set extraction: a reduced set of rules is extracted in the form: “IF contains <<SA>> THEN <<apply activity label>>”.

The obtained model should be checked on an external test set to validate it. Using the model requires applying the rules to the unknown molecule to produce the class label: the model tags the compound as toxic when one or more SAs are present, and as non-toxic if no SA is found. Dually, the user can ask SARpy to generate rules related to non-toxic substances, and use them to better assign molecules to the non-toxic class.

The SARpy tool is distributed in two different versions: W-SARpy is compiled for Windows and include a graphical user interface; the second is S-SARpy and is a python library without a graphical interface.

The functionalities of both version are the same.

Before starting, please check the prerequisites section and make sure your computer is compatible with SARpy. The second section will guide you trough the installation process; in the third and fourth you will work with dataset and ruleset respectively; the last one will explain you how to predict and validate the model.

If you publish a model created by SARpy please quote one of the following: [1], [2], [3].

Chapter 1

Prerequisites

The software and hardware prerequisites for W-SARpy are the following:

- Microsoft Windows XP SP1\SP2\SP3, Vista, 7, 8, 8.1, 10;
- Any Intel or AMD processor x86 or x64 (suggested Intel Core i5-750 or greater);
- 512MB RAM (1GB is suggested for faster computation);
- Up to 30MB available hard disk space (dataset files not included);
- Administrator rights are not needed for the installation process but suggested;
- A compression\decompression software (Windows built-in tool, WinZip, WinRar).

The software and hardware prerequisites for S-SARpy are the following:

- Any Linux distribution or OsX operating system;
- Any Intel or AMD processor x86 or x64 (suggested Intel Core i5-750 or greater);
- 512MB RAM (1GB is suggested for faster computation);
- Up to 2MB available hard disk space (dataset files not included);
- A compression\decompression software (7-zip, rar, tar);
- python 2.7 libraries;
- openbabel python modules.

It is a recommended best practice to back-up your system and data before you remove or install any software.

Chapter 2

Installing and starting SARpy

2.1 W-SARpy

The SARpy tool is distributed as a .ZIP package; this archive contains all the files the tool needs to work correctly. Once you have downloaded the package from the download area of the VEGA website (SARpy link), simply unpack it in any folder you like. To extract the files you can use the Windows integrated ZIP tool, or any file compression software you prefer (e.g. WinZip or WinRar).

Now enter in the folder just created and double click on the file "SARpy.exe"; the software will automatically starts showing its splash screen for a few seconds. After that the main window should appear, focused on the Dataset Managing Tab, as shown in 2.1.

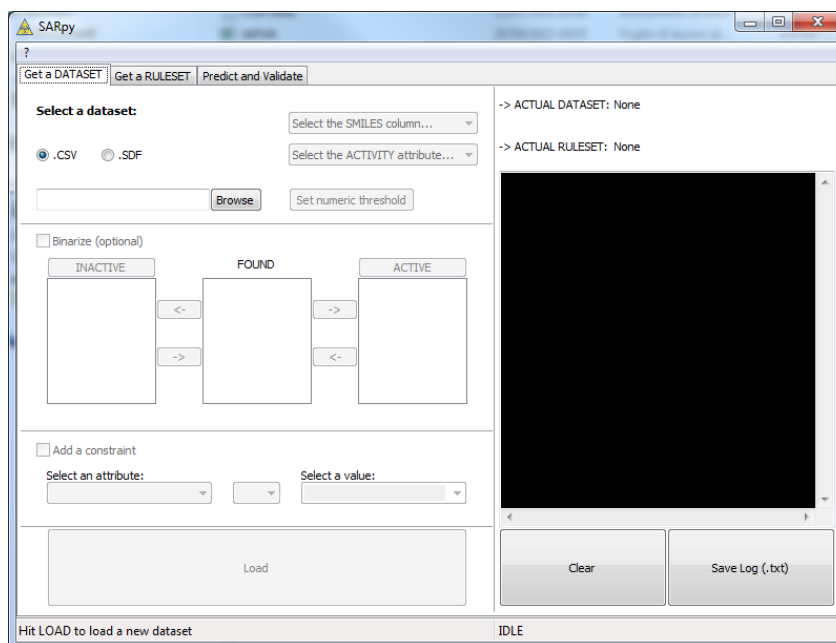


Figure 2.1: The main window of the SARpy tool.

2.2 S-SARpy

Please note that all the terminal commands reported here and in the following sections dedicated to Linux, are for Ubuntu and Ubuntu-based distribution. If you have a different distribution please check for the commands working for you.

For Linux based computer the SARpy tool is distributed as a tar.gz archive; you may first download it from the download area of the VEGA website (SARpy link). Now simply unpack it in any folder you prefer (we suggest you to select your home directory):

```
1 cd ~
2 tar -zxvf sarpy.tar.gz
3 cd sarpy
```

A “sarpy” folder will be created; you will find three files:

- SARpy.py, the main program;
- SARpytools.py, a library of functions;
- SARpyRun.py, a sample python script for using SARpy;
- ds.csv, a small dummy dataset for learning to use SARpy.

If you do not have Python 2.7 installed on your machine, please follow the following step by step before going on reading this user’s manual. Otherwise just ignore them.

```
1 sudo apt-get update
2 sudo apt-get upgrade
3 sudo apt-get install build-essential
4 sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3
   -dev tk-dev libgdbm-dev libc6-dev libbz2-dev
5 cd ~/Downloads/
6 wget http://python.org/ftp/python/2.7.5/Python-2.7.5.tgz
7 tar -xvf Python-2.7.5.tgz
8 cd Python-2.7.5
9 ./configure
10 make
11 sudo checkinstall
```

Next, if you do not have the openbabel chemical toolbox installed on your machine, please proceed installing it before starting SARpy (please refer to the openbabel website for files and instructions).

For using the SARpy tool now you can simply launch a python console typing the followings in the terminal:

```
1 cd ~/sarpy
2 python
3 import SARpy
4 import operator
```

The last two commands will enable you to call all the functionalities that the SARpy tool offers.

Chapter 3

Working with datasets

The SARpy tool allows you to work with datasets containing molecules with their respective properties; it is possible to load and manage a dataset, either for using it as training set to generate models, or as test set to validate models, or as a collection of untested molecules whose activity has to be predicted.

SARpy works only with a specific dataset format based on SMILES, which is automatically created by the system itself using data coming from external sources, such as text or excel files. All molecular structures contained within the selected external file are read, parsed and converted into the specific SARpy format, and then memorized as a SARpy dataset. This method assures reliability of the dataset and a short elaboration time.

The external data source must be either CSV or SDF files. Please be sure that your file meets all the format specification:

- **CSV format:** all the floating values must use the dot as decimal separator; as the CSV file requires each value must be separated by comma; the file must be column-wise (i.e. each column is a property); the first row of the file must contain property labels;
- **SDF format:** all the floating values must use the dot as decimal separator; each entry must be separated by the character sequence "\$\$\$\$"; each property name must be surrounded by "><" and ">" .

You can perform two main actions:

- **Loading** a new dataset from external files (see section 3.1);
- **Managing** and preparing the current dataset (see section 3.2).

3.1 Loading a dataset

W-SARpy

The Dataset Managing Tab is illustrated in Figure 3.1. To load the external file you simply select its format and then click on "Browse" button (Figure 3.2); if the operation is successful all the other functionalities in this tab should now be available. .

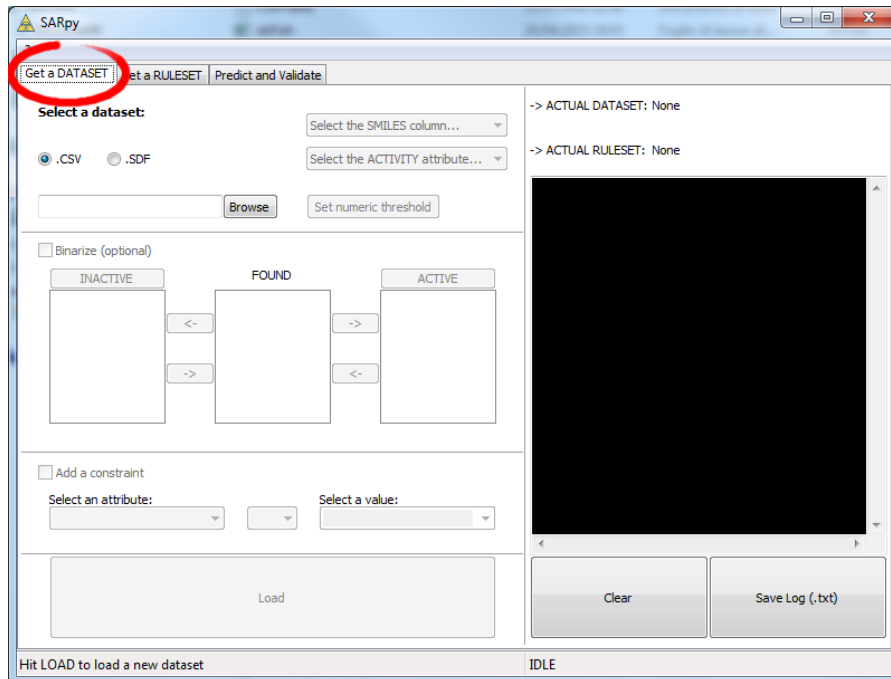


Figure 3.1: The figure shows the Dataset Managing Tab; here you can load and customize a dataset.

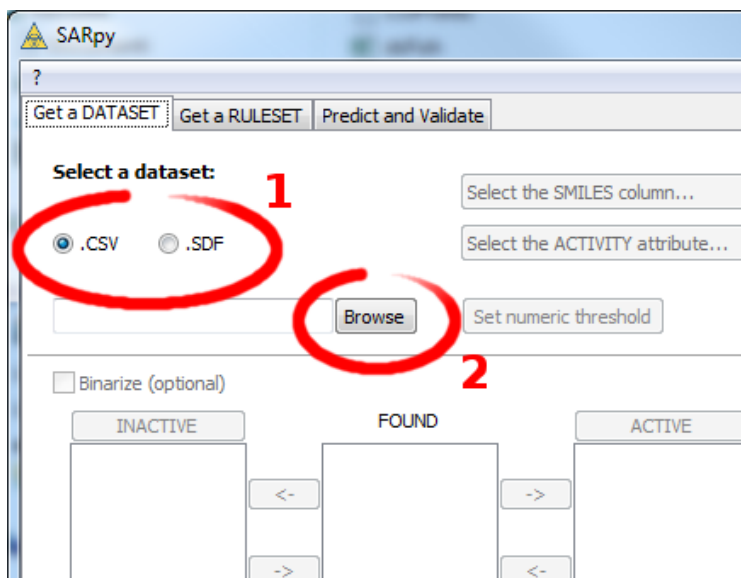


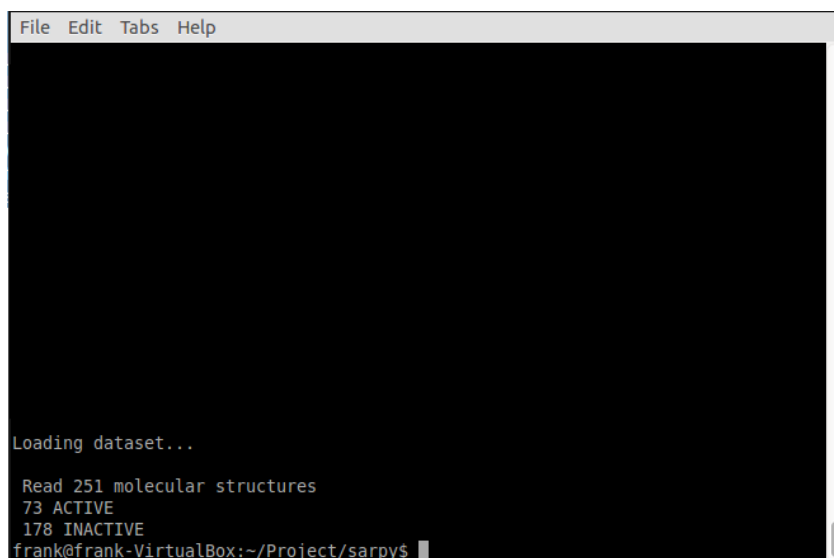
Figure 3.2: How to load an external file: first select the file format and then browse for it on your computer

S-SARpy

To load a new dataset you have to type in a python console the following:

```
1 dataset = SARpy.loadDataset('fileName.ext', 'ext', dictionary, 'SMILE-Key',  
                             filter);
```

where “fileName.ext” is the name of the file to be loaded, with its extension, “ext” can be either “CSV” or “SDF” and “SMILE-Key” is the attribute in the dataset referring to the SMILE of the molecule; dictionary and filter are presented in the next two sections. The function “loadDataset” will return you a reference to the complete dataset of molecules loaded, binarized and filtered from the external file.



```
File Edit Tabs Help  
  
Loading dataset...  
Read 251 molecular structures  
73 ACTIVE  
178 INACTIVE  
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 3.3: After the dataset has been correctly loaded you should read in the shell the total number of molecules.

3.2 Managing the current dataset

Once the set of molecules has been correctly loaded from external file, you can manage it according to your own needs. There are two options:

- **Binarize** the current dataset;
- **Filter** the current dataset.

Even if SARpy may properly work using a non-binary classification (i.e. considering more than two activity classes), a lot of case studies divide the compounds using a binary classification scheme, generally labeling each compound with the generic “Active” or “Inactive” labels. The binarization tool provides you with the capability to create this kind of classification from a multi-class one, relabeling the classes. The binarization operation requires you to specify which of the present classes are to be considered as active and which

are instead considered as the inactive ones. Once the binarization operation is done, all molecular structures will change their activity description according to the new parameters.

The binarize tool also works with datasets that use continuous values; in this case a proper threshold value within the range must be specified in order to split the set into "Active" and "Inactive" molecules. Please refer to the Chapter 6 for details about this functionality.

The second possible action provided in SARpy is for filtering data. Testing methodologies often need to split a datasets into several subsets, each with a particular property: a classical example is the division in training and testing sets, the first dedicated to the generation of the model, the second used only for validating the model. This constraint tool allows you to apply one or more constraints on the whole dataset, splitting it into several parts, and obtaining a reduced dataset that meet the given restrictions.

W-SARpy

You can perform all these actions through the "Dataset Managing Tab"; once your data are ready, you click on the "Load" button in the bottom of this tab so that SARpy can create its own internal representation of the dataset (Figure 3.4). If it has been correctly created, the filename of the external data source and the total structure count should be reported on the right, just above the "Info Dialog" (Figures 3.5).

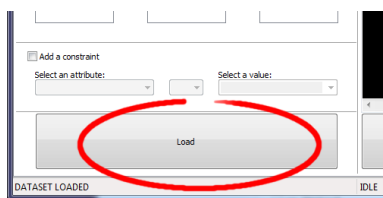


Figure 3.4: Once you are ready with your dataset, just click on the "Load" button you find in the bottom of this tab.

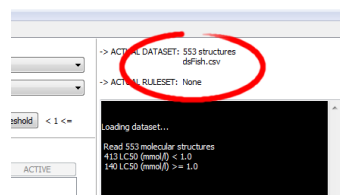


Figure 3.5: The image show the Info Panel, useful to check for error on every load operation.

If you need to binarize the dataset, in the central section of the tab check the “Binarize” checkbox, and move the labels into the right side with the provided buttons (Figure 3.6).

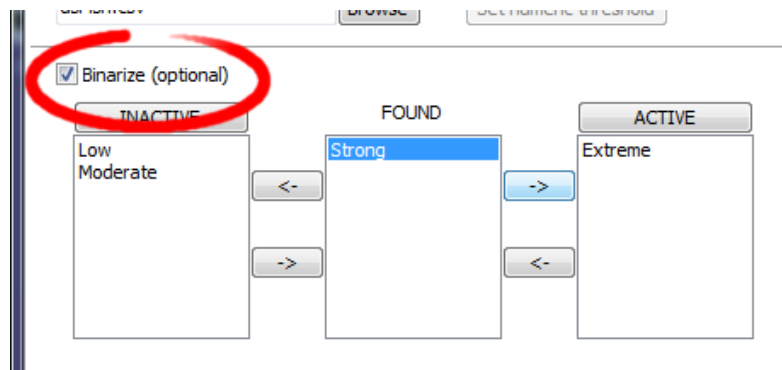


Figure 3.6: To binarize a dataset check the highlighted checkbox and move all the labels in the right new class.

Lastly, the bottom section is dedicated to filtering data; check the “Add a constraint” checkbox and compose a rule: select the attribute by the first drop-down list, the operator and then specify the value you need (Figure 3.7).

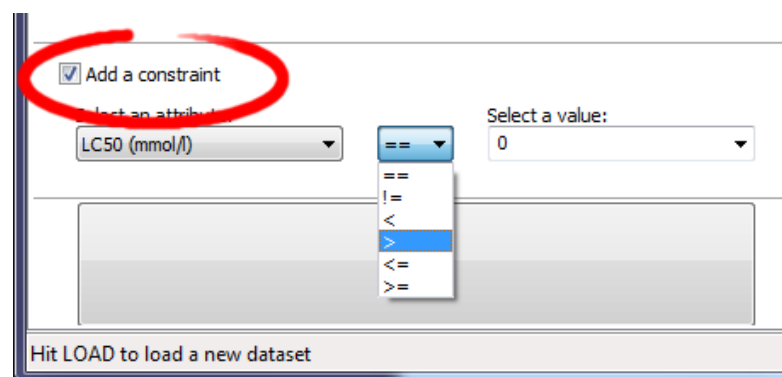


Figure 3.7: Here is the filtering tool; select the property you need to filter by and compose the filtering rule.

S-SARpy

For binarize data you first have to create binarization rules, and collect them in a sort of a dictionary:

```

1 filter1 = SARpy.Filter('keyName', 'Value', operator);
2 filter2 = SARpy.Filter('keyName', 'Value', operator);
3 ...
4 filterN = SARpy.Filter('keyName', 'Value', operator);
5 dict = {'CLASS_A':filter1, 'CLASS_A': filter2, 'CLASS_B':filterN};

```

where “keyName” is the attributes in your dataset you want to binarize your data by, “Value” is the desired target value for the attribute, and operator is defined as a python standard operators as functions (see python reference for further details).

For example you can define a rule like the following one:

```
1 filt = SARpy.Filter('CLASS', 'Low',operator.eq);
```

and then add it to a dictionary:

```
1 dict = {'INACTIVE':filt};
```

This rule indicates that every molecule with a value of the attribute “CLASS” equals to “Low” will be classified as “Inactive”.

In a similar way you can create filters for considering only a subsets of the initial full dataset; note that you do not have to generate a dictionary:

```
1 filt = SARpy.Filter('ID', 200, operator.gt);
```

This rule means that you want to consider only entries of your dataset with an ID greater than 200.

Chapter 4

Working with rulesets

SARpy's Ruleset Managing Tab allows you to create a ruleset, that is a list of rules that establishes relationships among various substructures and the selected activity classes. These rules are written using the SMILES format for the chemical structures, and always follow this syntax:



This syntax indicates that the selected fragment usually identifies the activity of a molecule as Developmental toxicant, with a likelihood ratio of 1.06. SARpy's ruleset models therefore molecular activity, and it must be generated taking into account a wide range of conditions to improve its overall reliability.

Unlike the datasets, that are only loaded into the system starting from an external file, a ruleset might be saved and loaded for further analysis on the same endpoint. These models are saved by SARpy in a user specified folder, in plain TXT text format, and are easily interpretable and ready-to-use for scientific publications.

SARpy has two ways of creating models; these are similar in the produced result and in the way they operate, but generate models that have different purposes. You can use SARpy as:

- A classifier to **predict** a property: SARpy will generate a model that establishes relationships between each found substructure and all the activity classes specified during the dataset loading operation. Models generated by this modality consist of a list of rules that might be used to classify other molecules into one of the considered classes. Since the purpose is to predict the activity of new structures, rules generated in this modality are generally more detailed than rules produced in the Extractor mode.
- A knowledge extractor tool to **extract** relevant substructures: the SARpy tool tries to generate some new knowledge from the current dataset, analyzing it versus a singular activity class and establishing relationships among substructures and the specified activity class. This modality produces a list that is somewhat less specific than that produced in the other mode, but gives information about substructures that are possibly related to a specific activity.

Generally speaking, each SAR model works better for molecules that have similar characteristics; this means that SAR models might always be tailored around the specific batch

of structures being used to create it. So, a minute regulation of a certain number of parameters is generally possible for SAR model generation. SARpy is not an exception, as it provides some functionalities that affects robustness, sensitivity and sensibility of the developed models.

Through the user interface you can regulate models precision and also the definition of the structural alerts:

- **Model precision:** SAs extracted by SARpy are usually associated with a value that defines their precision; you can regulate the level of sensitivity and specificity by various parameters that affect the alert precision. For a quick tuning there is an “Auto” setting, by which you just have to select among three predefined values of precision: “min” means a more “sensitive” result (i.e. it minimizes the unpredicted rate) while “max” will produce a more “specific” result (i.e. it minimizes the error rate). In alternative, using the “Manual” regulation mode you can set the minimum likelihood ratio you prefer for each structural alert. An increase in this parameter corresponds to a higher precision of the model.
- **Structural Alerts Options:** the process of creating a new model implies the analysis of each molecular structure and the fragmentation of molecules into several substructures; these last are matched with the information about activity in order to establish any possible relationship between substructures (i.e. structural alerts, SAs) and activities. SARpy gives you the capability to define some parameters of the SAs: the maximum and the minimum number of atoms each SA is composed of (which affect the number of SAs considered and the computation time) and the minimum number of occurrences needed for each SA to be considered as valid (higher values correspond to more precision).

W-SARpy

The general appearance of “Ruleset Managing Tab” is shown in Figure 4.1.

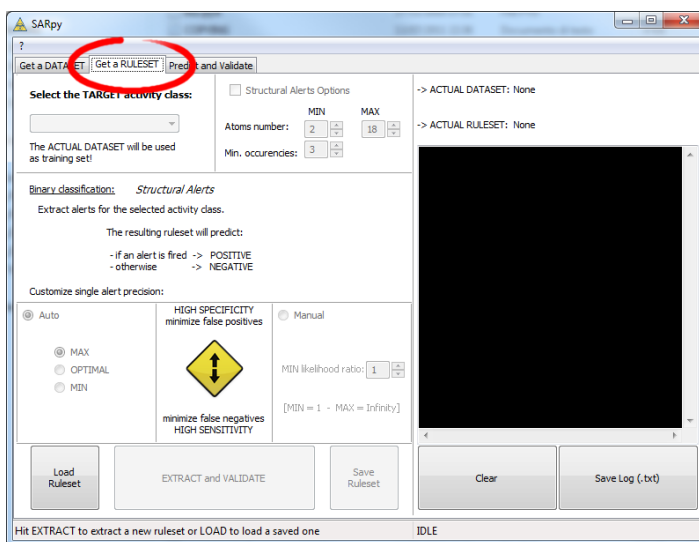


Figure 4.1: The Ruleset Managing Tab.

To load or save a ruleset, you have to click on the dedicated buttons in the bottom of the tab and follow the instruction written in the dialog window that will open. Otherwise, to create a brand new model you must have a dataset loaded and you also have to specify several parameters of the model. The loaded dataset must be valid and contain at least two activity classes: if something does not meet these prerequisites all the options in this tab will not be enabled.

The first parameters you have to regulate regard the structure of the SAs, such as the minimum and maximum of the number of atoms (Figure 4.2); read the previous section for details about this setting.

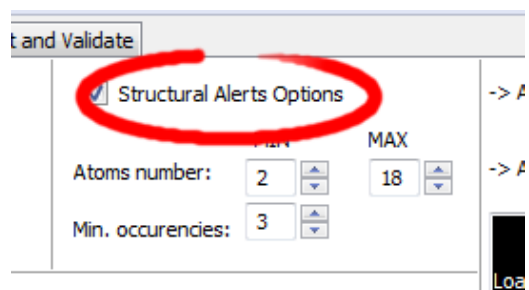


Figure 4.2: After you have checked the "Structural Alerts Options" , it is possible to modify parameters regarding the structure of the SAs.

Then you have to set the precision you want for the model (Figure 4.3); read the previous section for details about this setting.

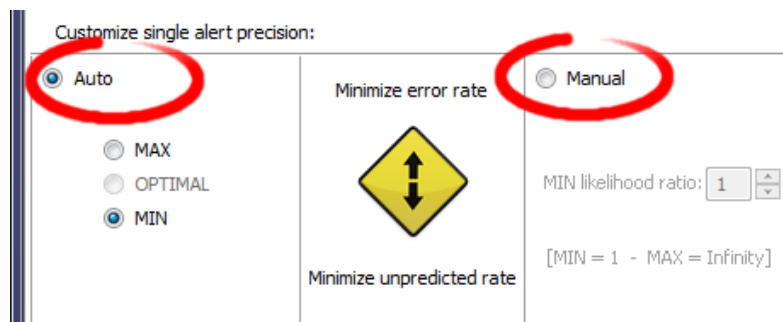


Figure 4.3: From this panel it is possible to regulate the precision of the model to be generated.

After every parameters is set up correctly, click on the button "Extract and Validate" to generate the model (Figure 4.4); the info panel will show you the progress of the computation.

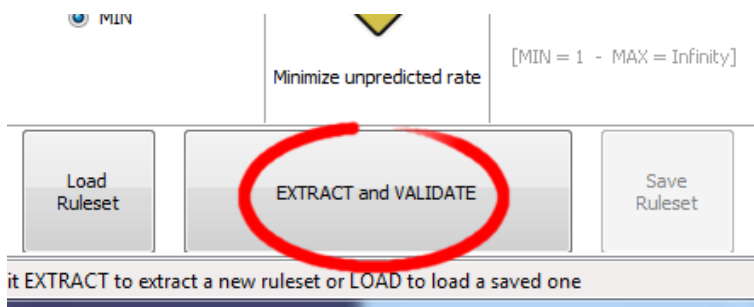


Figure 4.4: Once every desired properties of the model have been set, click on the "Extract and Validate" button to generate the model.

4.1 S-SARpy

To load an existing ruleset you simply call the `loadSmarts` function, specifying the name of the `.txt` file; this function will return you a pointer to the collection of rules.

```
1 rules = SARpy.loadSmarts('fileName.txt');
```

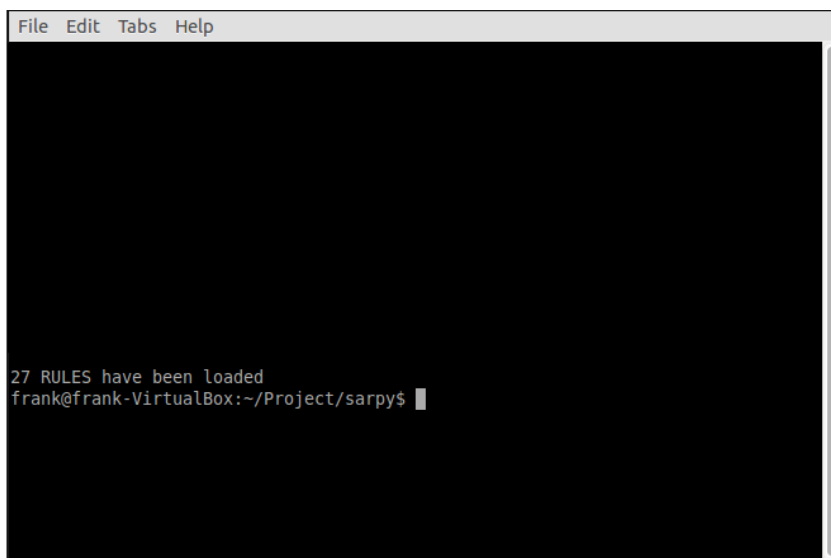


Figure 4.5: After you have called the "loadSmarts" function you should read in the shell the number of rules loaded.

Otherwise, if you need to create a new ruleset from scratch, first you have to fragmentize your dataset:

```
1 SARpy.fragmentize(dataset, minAtoms, maxAtoms, 'target');
```

where `minAtoms` and `maxAtoms` are mandatory and represents the minimum and maximum number of atom for each SA, and "target" is an activity class (leave empty or specify "None" for considering every class).

```
File Edit Tabs Help

Fragmenting...

887  substructures found...
1136 substructures found...
1192 substructures found...
926  substructures found...
483  substructures found...
187  substructures found...
31   substructures found...
1    substructures found...
0    substructures found...

FRAGMENTS: 4843

Evaluating fragments on the training set...

-> elapsed time: 11.56 seconds
    fragmentation 11.28 seconds
    matching 0.28 seconds
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 4.6: The result of the fragmentation in the shell

Now you can extract rules in the following way:

```
1 rules = SARpy.extract(dataset, minHits, minLR, minPrecision, 'target');
```

You have to select the minimum hits value (i.e. the minimum number of occurrences of a SA), the minimum likelihood ratio, the minimum precision of the model and the “target” activity class. Default values for each parameter is respectively 3, 1, “None” and “None”; all attributes but dataset are optional and you can specify the value for each single parameter like this:

```
1 rules = SARpy.extract(dataset, minLR=2);
```

```
File Edit Tabs Help

Extracting rules...

2427 ACTIVE substructures
233 of which are potential alerts

2995 INACTIVE substructures
189 of which are potential alerts

Extracted:
12  ACTIVE
15  INACTIVE

RULES: 27

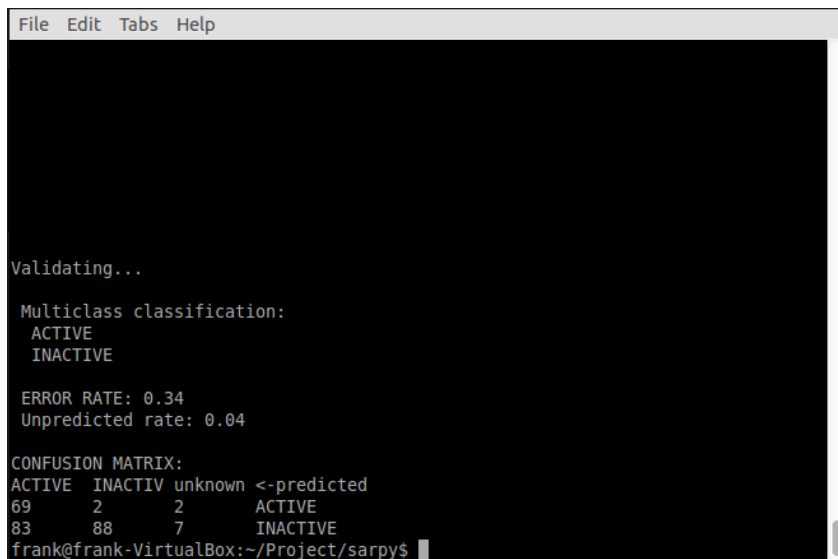
-> time: 0.20 seconds
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 4.7: When all the rules are extracted the number of generated rules are shown in the shell

You can now validate your ruleset against your dataset in the following way:

```
1 SARpy.validate(dataset);
```

This command will print several information about the validity of the generated classifier, like the confusion matrix and the precision; see figure 4.8



```
File Edit Tabs Help

Validating...

Multiclass classification:
ACTIVE
INACTIVE

ERROR RATE: 0.34
Unpredicted rate: 0.04

CONFUSION MATRIX:
ACTIVE  INACTIV  unknown  <-predicted
69      2         2         ACTIVE
83      88        7         INACTIVE
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 4.8: The results of the validation process in the shell.

You can save the ruleset to a .txt file for further analysis calling the “saveSmarts” function, specifying the ruleset and the “filename” you like.

```
1 SARpy.saveSmarts(rules, 'fileName.txt');
```

The output will be a .txt file like the following (Figure 4.9):

```
SMARTS Target Training LR
CCCCCCCCC ACTIVE inf
c1cc(Cl)ccc10 ACTIVE inf
CCCC0Cclcccc1 ACTIVE inf
0c1c(C(C)C)cccc1 ACTIVE inf
c1(C=0)ccc(0)cc1 INACTIVE 4.10
C(=0)CCCC INACTIVE 1.85
c1cc(C#N)ccc1 INACTIVE inf
Cc1c(N)cccc1 INACTIVE inf
CN(C)C INACTIVE inf
CN(C)c1cccc1 INACTIVE inf
c1c(F)cccc1 INACTIVE inf
c1cccc1C(=0)0C INACTIVE 1.23
```

Figure 4.9: An example of the generated .txt file containing all the extracted rules

Chapter 5

Predicting and validating

The last tool is for predicting activities of unseen compounds (i.e. apply the ruleset on the dataset) or for validating the model.

Once the prediction process has correctly finished you can save the result. A plain text file will be generated: the prediction information is row-wise, meaning that each row is the prediction of a specific compound found in the dataset. Each row contains, in this order, the following information:

- The compound SMILE;
- The prediction as a label (standard values are “Active” and “Inactive”);
- The likelihood-ratio test value;
- The SMART.

Optionally it is possible to add in this text file other information among those contained in the dataset.

The last step you can perform is model validation, that will give you information about the model performance and will check the accuracy of the model’s representation of the real system.

5.1 W-SARpy

This last section of the SARpy tool is the Prediction and Validation Tab; once you have correctly loaded a dataset in the system and created or loaded a set of rules (as seen in the two previous sections), through this tab you can predict the activity of other unseen compound (i.e. apply the ruleset on the dataset) and also validate the model. This tab is represented in figure 5.1.

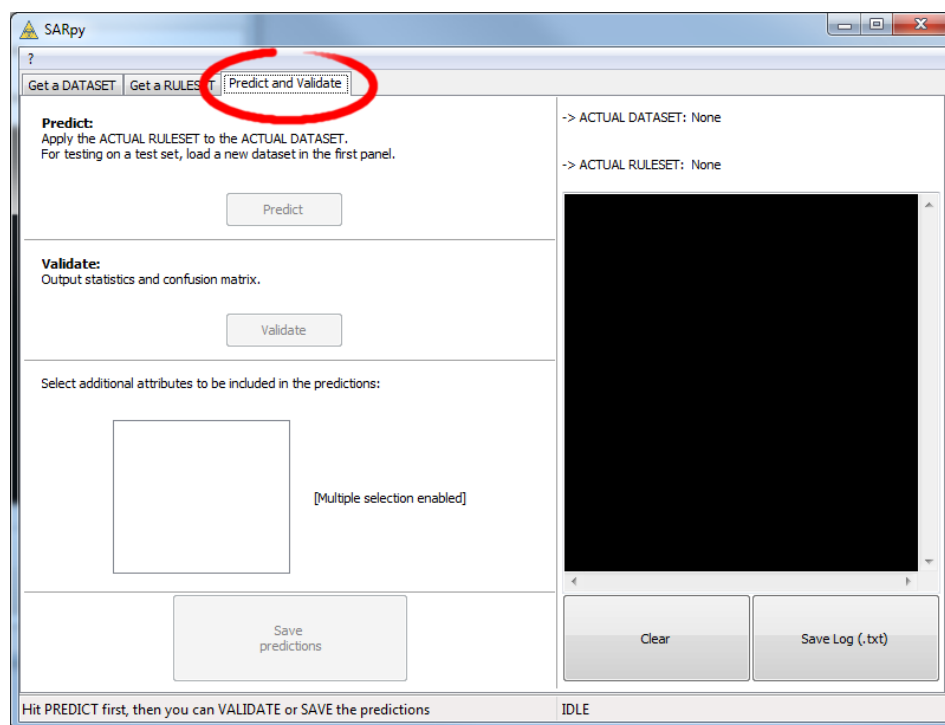


Figure 5.1: The Predict and Validate Tab.

To predict the activity of the compounds listed in the testing dataset you simply have to click on the "Predict" button you find in the top of this tab. The process will be quite fast and you can check whether it has finished reading the info panel on the right: when the process is over you should read "xxx structured matched", as shown in figure 5.3.

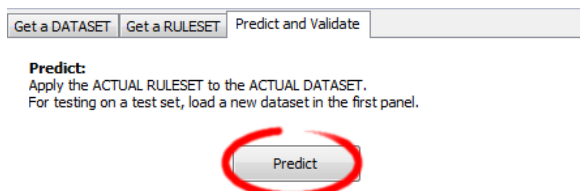


Figure 5.2: To start the prediction process click on the "Predict" button you find in the top of this tab.

To start the validation of the generated model on the loaded dataset just click on "Validate" button you find in the middle of this tab; as the process finish, the info panel will give you the relevant information (i.e. the error rate and the confusion matrix) regarding the model performance (see figure 5.5).

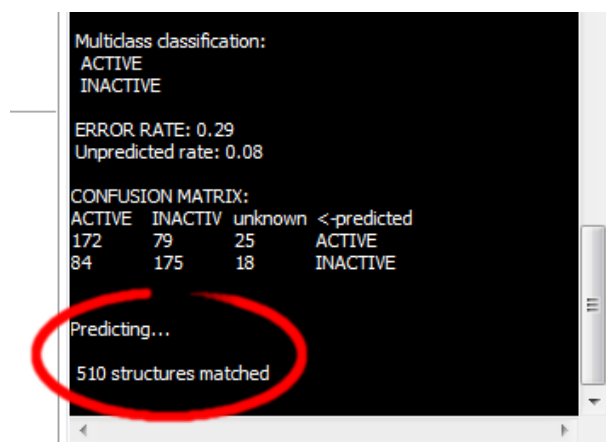


Figure 5.3: The info panel will show you when the prediction process is over.

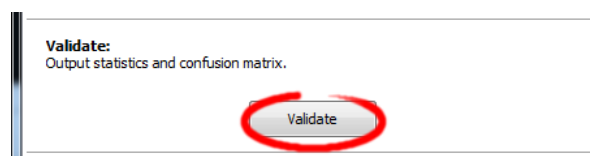


Figure 5.4: Click on the validate button to start the model validation process.

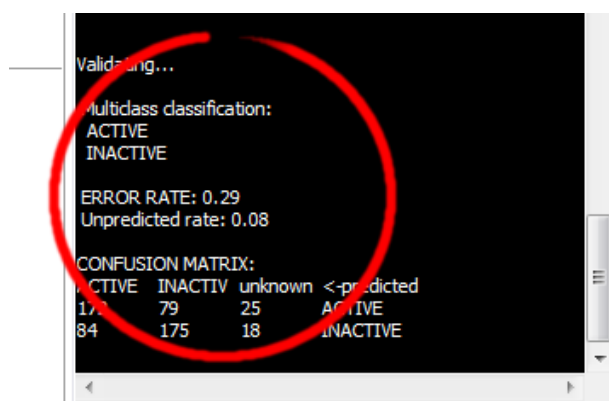


Figure 5.5: As the validation process is over the info panel will show its result.

Optionally it is possible to add final .txt file other information among those contained in the dataset; to do this just select those you want to add from the list before saving the prediction result (see figure 5.6).

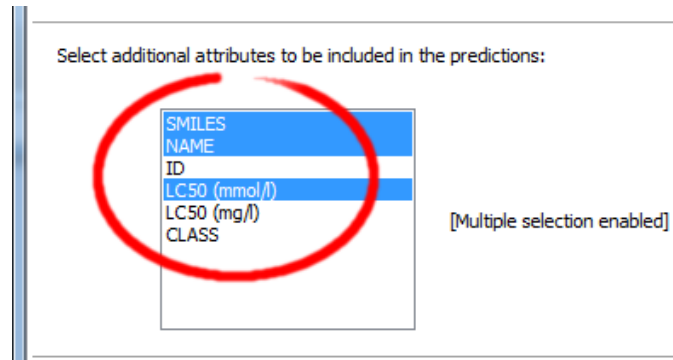


Figure 5.6: Select from this list all the attributes you would like to add to the prediction result.

5.2 S-SARpy

To predict the activity of various compounds you have to use the “predict” function:

```
1 SARpy.predict(ruleSet, dataSet);
```

The process will be quite fast and you can check whether it has finished reading the printed information: when the process is over you should read “xxx structured matched” (see Figure 5.7).

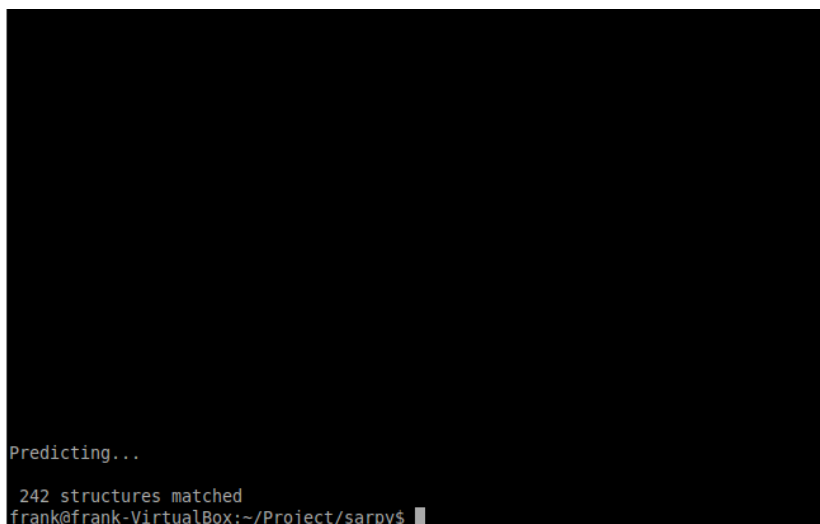


Figure 5.7: As the predicting process is over, the number of matched structures will be displayed in the shell.

You can now validate the generated model with respect to a dataset by the “validate” function; the shell output is shown in figure 4.8.

```
1 SARpy.validate(dataSet);
```


For further analysis you can save the predictions just elaborated to a plain .txt file:

```
1 SARpy.savePredictions(dataset, 'filename', keys, 'separator');
```

where “keys” is a vector of names of attributes defined as $[key'_1, key'_2, \dots, key'_n]$ and “separator” is a character or special character (such as ‘\t’ for tab, the default value) that will be used for separating values of each attributes in the final output file.

For example the following command:

```
1 SARpy.savePredictions(dataset, 'predictions.txt', sep=',');
```

will create the file “predictions.txt” like the following one (Figure 5.8):

```
SMILES,Prediction,Training LR,SMARTS,CLASS,ID
c1c(C(=O)CBr)c(OC)ccc1OC,ACTIVE,1.72,0c1cccc1,Extreme,462
CC1(C)Oc2c(OC(=O)NC)cccc2C1,ACTIVE,1.72,0c1cccc1,Extreme,467
0c1c(Cl)cc(Cl)c(Cl)c1Cc1c(Cl)c(Cl)cc(Cl)c10,ACTIVE,inf,c1cc(Cl)ccc10,Extreme,47
Nc1ccc(F)cc1,INACTIVE,inf,c1c(F)ccc1,Moderate,337
CCOP(=S)(OCC)Oc1nc(Cl)c(Cl)cc1Cl,ACTIVE,9.75,P(=S)(OCC)O,Extreme,513
C=C(CCl)CCL,ACTIVE,3.55,C=C,Extreme,477
c1c(Cl)c(O)c(Cl)cc1C#N,INACTIVE,inf,c1cc(C#N)ccc1,Moderate,478
n1c(Cl)c(Cl)c(Cl)c(Cl)c1Cl,None,,Extreme,490
c1(C=O)cc(C(F)(F)F)ccc1,ACTIVE,1.12,c1cccc1,Extreme,342
Nc1c(Cl)c(Cl)cc(Cl)c1Cl,ACTIVE,5.69,c1c(Cl)cccc1Cl,Extreme,519
```

Figure 5.8: The prediction file generated by the “savePredictions” function.

Chapter 6

W-SARpy step-by-step

This last section will guide you through the W-SARpy/S-SARpy tool step-by-step, performing each action needed to load a dataset, to create or load a ruleset and to predict and validate a result.

The toy example here proposed is based on a public dataset from the Environmental Protection Agency (EPA), Mid-Continent Ecology Division, Duluth (MN). All toxicity data were retrieved from [4] and checked in the DEMETRA project.

Important note for user: this is only a toy-example about how to use the W-SARpy/S-SARpy tool; we do not want to build a real model.

6.1 Preparing the CSV dataset file

The first thing you need before opening the tool is a .csv or .sdf file containing the dataset of compounds. Here we will start from an .xls file, that you can open both with Microsoft Excel or LibreOffice or OpenOffice (or equivalents); the file must be column-wise, i.e. each column represents a property you have in the dataset (SMILE, Name, ID, class and so on) and the first row of the file must specify the property label (see figure 6.1). Please also note that floating point values must have the dot as decimal separator.

1	2	3	4	5	6
ID	NAME	SMILES	LC50 (mg/l)	LC50 (mmol/l)	CLASS
1	4-(HEXYLOXY)-HI-ANISALDEHYDE	c1cc(C=O)cc(OC)c1OCCCCC	2.67000	0.0112987	Low
2	5-BROMO-2-NITROVANILLIN	c1(OC)c(N(=O)=O)c(C=O)cc(Br)c1O	73.30000	0.2655412	Moderate
4	P-CHLOROPHENYL-O-NITROPHENYL ETHER	c1cc(Cl)ccc1Oc2c(N(=O)=O)ccc2	1.92000	0.0076908	Strong
5	3-CHLORO-O-FORMOTOLUIDIDE	Cc1c(NC=O)cccc1Cl	46.60000	0.2747480	Low
6	DI-n-BUTYLISOPHTHALATE	CCCCOC(=O)c1cccc(C(=O)OCCCC)c1	0.90000	0.0032333	Moderate
7	1,1-DIPHENYL-2-PROPYN-1-OL	C(c1ccccc1)(c2ccccc2)O)C#C	11.10000	0.0532988	Moderate
8	4,7-DITHIADECANE	CCSCCCSCCC	7.52000	0.0421643	Strong
9	4,9-DITHIADECANE	CCSCCCSCCC	2.99000	0.0144857	Low
11	2-CHLOROETHYL-N-CYCLOHEXYL CARBAMATE	C1CCOC(=O)N(C1)CCCC1	35.00000	0.1701673	Extreme
11	PHENOBARBITAL	N1C(=O)C(c2ccccc2)(CC)C(=O)NC1=O	484.00000	2.0838715	Low
12	2,4-DINITROPHENOL #9	Oc1c(N(=O)=O)cc(N(=O)=O)cc1	13.30000	0.0722394	Moderate
13	URETHANE	O=C(N)OCC	5240.00000	58.8169267	Extreme
14	BENZAMIDE	c1ccccc1C(=O)N	661.00000	5.4564966	Moderate
15	1,1-DIMETHYLHYDRAZINE	NN(C)C	7.85000	0.1306156	Low
16	AMOBARBITAL	N1C(=O)C(CC)(CC(C)C)C(=O)NC1=O	85.40000	0.3773585	Strong
17	CAFFEINE	CN1C=NC2=C1C(=O)N(C(=O)N2C)C	151.00000	0.7774688	Extreme
18	2-METHYL-1,4-NAPHTHOQUINONE	c1ccc2c(C(=O)C)c(C(=O)C)cc12	0.11000	0.0006389	Low
19	2,3,4,6-TETRACHLOROPHENOL	Oc1c(Cl)c(Cl)c(Cl)cc1Cl	1.03000	0.0044418	Low
20	4-CHLORO-3-METHYL PHENOL #1	Oc1cc(Cl)c(C)cc1	7.38000	0.0517568	Low
21	DIETHYL ETHER	O(C)CC	2560.00000	34.5385861	Moderate
22	ANILINE #1	Nc1ccccc1	134.00000	1.4388489	Low
23	CARBARYL (SEVIN) #2	CN(C=O)Oc1cccc2ccccc12	8.93000	0.0443771	Extreme
24	ETHANOL	CCO	14200.00000	308.2266117	Strong
25	2-HYDROXYBENZAMIDE	c1(O)cccc1C(=O)N	101.00000	0.7364737	Extreme
26	HEXANAL #2	CCCCCC=O	14.00000	0.1397764	Low

Figure 6.1: The example dataset loaded as Excel file.

From your spreadsheet software you can export a CSV file: simply click on “Save as...” and select the .csv extension from the drop-down list you can find under the file name (see figure 6.2)

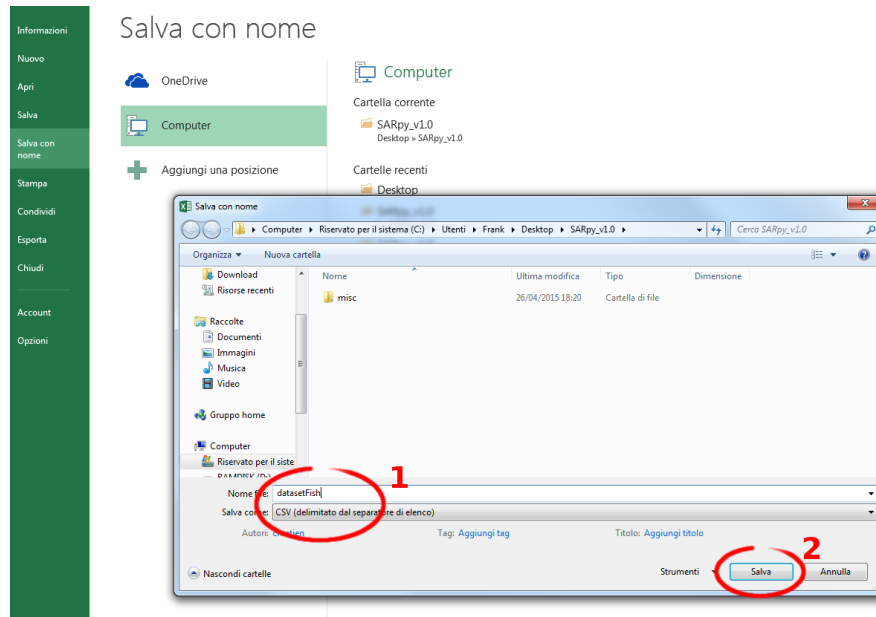


Figure 6.2: To export your .xls to .csv in Microsoft Excel simply save the file and choose the .csv extension from the list.

It can happen that an information message stating that you will lose some file features appears; click on “Yes”.

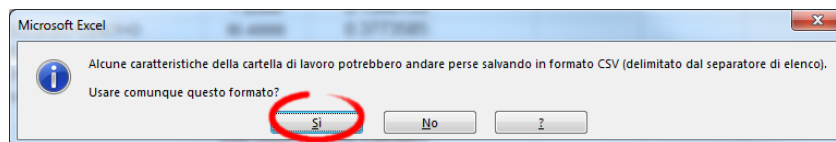


Figure 6.3: Click on “Yes” to safely close the message and continue.

Now that your CSV file is ready, open it with a text editor; check that your data are not corrupted and also that the value separator is a comma (if not, you have to set the correct separator in the Control Panel/System Setting, an option generally listed in the language section). The file should look like the one showed in figure 6.4.

```

datasetFish - Blocco note
File Modifica Formato Visualizza ?
ID,NAME,SMILES,LC50 (mg/1),LC50 (mmo1/1),CLASS
1,4-(HEXYLOXY)-M-ANISALDEHYDE,c1cc(C=O)cc(OC)c10CCCCC,2.67000,0.0112987,Low
2,5-BROMO-2-NITROVANILLIN,c1(OC)c(N(=O)=O)c(C=O)cc(Br)c10,73.30000,0.2655412,Moderate
4,P-CHLOROPHENYL-O-NITROPHENYL ETHER,c1cc(C1)ccc1oc2c(N(=O)=O)cccc2,1.92000,0.0076908,Strong
5,3'-CHLORO-O-FORMOTOLUIDIDE,c1c(NC(=O)cccc1C,46.60000,0.2747480,Low
6,DI-n-BUTYLISOPHTHALATE,CCCC(=O)C1CCCC(C(=O)O)CCCC1,0.90000,0.0032333,Moderate
7,"1,1-DIPHENYL-2-PROPYN-1-OL",C(C1CCCC1)(C2CCCC2)O)C#C,11.10000,0.0532988,Moderate
8,"4,7-DITHIADECANE",CCSCCSCCC,7.52000,0.0421643,Strong
11,"4,9-DITHIADECANE",CCSCCSCC,2.99000,0.0144857,Low
14,2-CHLOROETHYL-N-CYCLOHEXYL CARBAMATE,C1COC(=O)NC1CCCC1,35.00000,0.1701673,Extreme
15,PHENOBARBITAL,N1C(=O)C(C2CCCC2)(CC(C(=O)N)C1=O,484.00000,2.0838715,Low
16,"2,4-DINITROPHENOL #9",Oc1c(N(=O)=O)cc(N(=O)=O)cc1,13.30000,0.0722394,Moderate
17,URETHANE,O=C(N)OCC,5240.00000,58.8169267,Extreme
19,BENZAMIDE,C1CCCC1C(=O)N,661.00000,5.4564966,Moderate
21,"1,1-DIMETHYLHYDRAZINE",NN(C)C,7.85000,0.1306156,Low
23,AMOBARBITAL,N1C(=O)C(CC)(CCC(C)C(=O)N)C1=O,85.40000,0.3773585,Strong
24,CAFFEINE,CN1C(=O)N(C)C(=O)C2=C1N=CN2C,151.00000,0.7774688,Extreme
25,"2-METHYL-1,4-NAPHTHOQUINONE",c1ccc2c(=O)c(c)cc(=O)c12,0.11000,0.0006389,Low
26,"2,3,4,6-TETRACHLOROPHENOL",Oc1c(C1)C(C1)C(C1)CC1,1.03000,0.0044418,Low
27,4-CHLORO-3-METHYL PHENOL #1,oc1cc(C)C(C1)CC1,7.38000,0.0517568,Low
30,DIETHYL ETHER,O(C)CC,2560.00000,34.5385861,Moderate
32,ANILINE #1,Nc1ccccc1,134.00000,1.4388489,Low
33,CARBARYL (SEVIN) #2,CNC(=O)Oc1cccc2cccc12,8.93000,0.0443771,Extreme
34,ETHANOL,CCO,14200.00000,308.2266117,Strong
36,2-HYDROXYBENZAMIDE,c1(O)CCCC1C(=O)N,101.00000,0.7364737,Extreme
38,HEXANAL #2,CCCCCC=O,14.00000,0.1397764,Low

```

Figure 6.4: Edit the file .csv just saved and verify that it meets all the prerequisites.

6.2 W-SARpy

With the .csv file ready you can open W-SARpy. A splash screen should appear and after a few second the main window should open, focused on the Dataset Managing Tab (Figure 6.5).

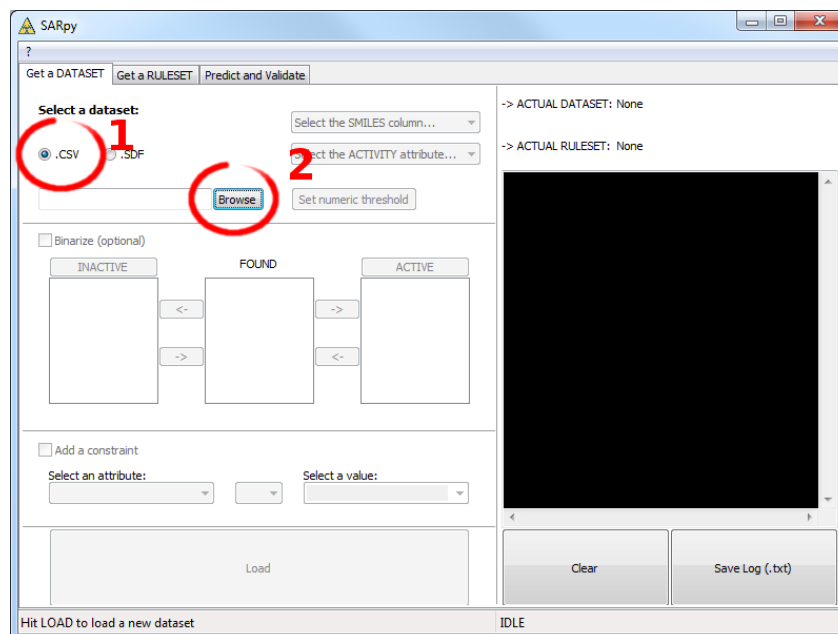


Figure 6.5: The W-SARpy tools as it opens.

Loading the dataset

The first thing you have to do is to load the .csv external file, only after having selected the file format you are about to use; in this example we select “.csv”. Click then the “Browse” button to search for the file in your computer (figure 6.6).

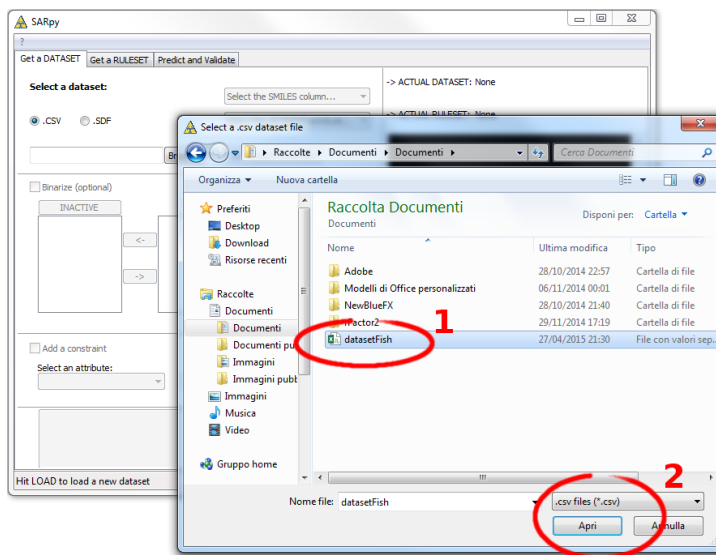


Figure 6.6: Look for the .csv file and click on load; if you cannot find it make sure the file format filter is set correctly.

Now choose from the first dropdown list (on the right) the name of the attribute in the dataset corresponding to the SMILES, so that W-SARpy knows which column contains the compounds SMILE (see figure 6.7).

Secondly, indicate to the software which column contains the activity attribute. In our example dataset we have a column called “class”, which specifies the class for each compound. Our classes are just string values, namely “Low”, “Moderate”, “Strong” and “Extreme” (see figure 6.8).

The activity attributes may also be a numeric value; in this case W-SARpy will let you know that you probably want to set a threshold for binary classification of compound. Click “Ok” button; a new pop-up will be displayed asking you this numeric threshold (figures 6.9 and 6.10).

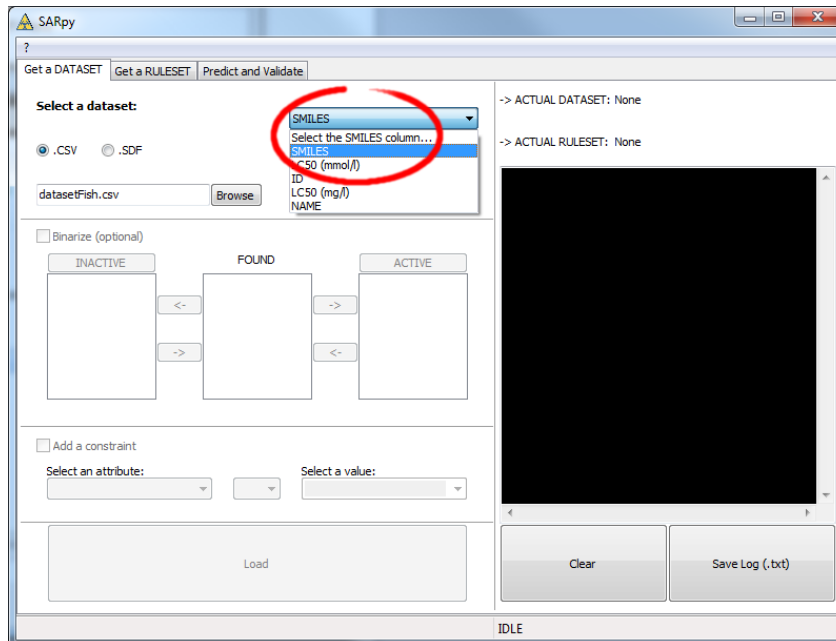


Figure 6.7: From the list select the attributes for SMILES.

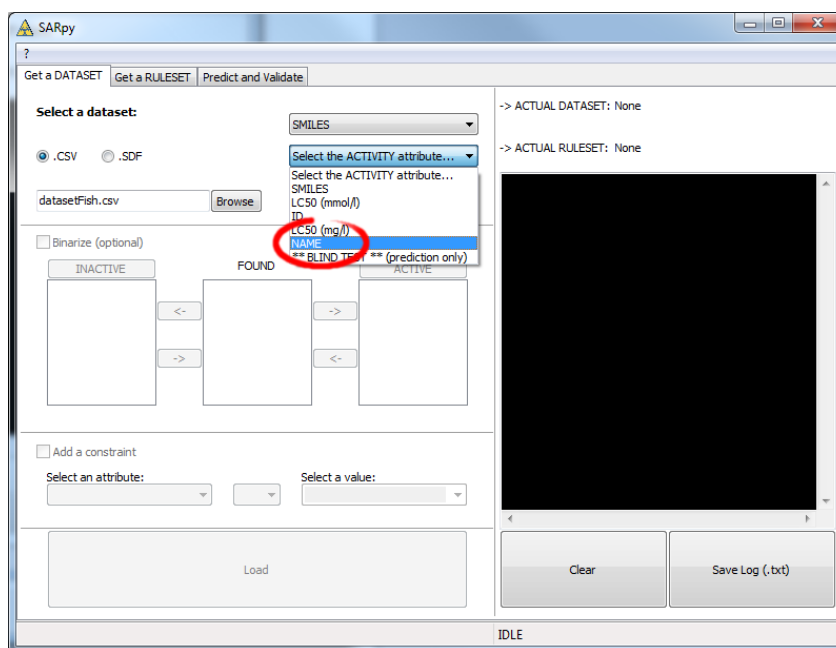


Figure 6.8: Select the activity attribute.

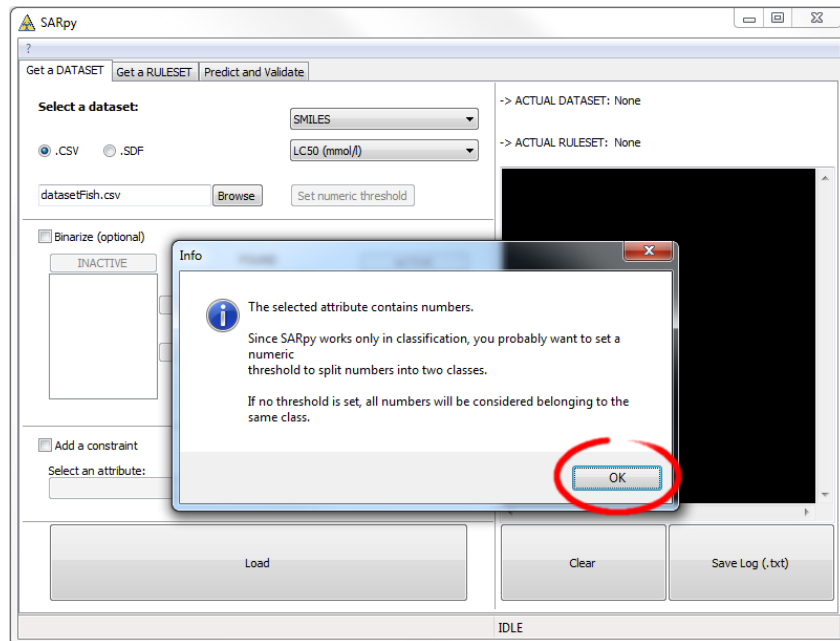


Figure 6.9: A popup will warn you if you select a numeric activity attribute. Just click “Ok”

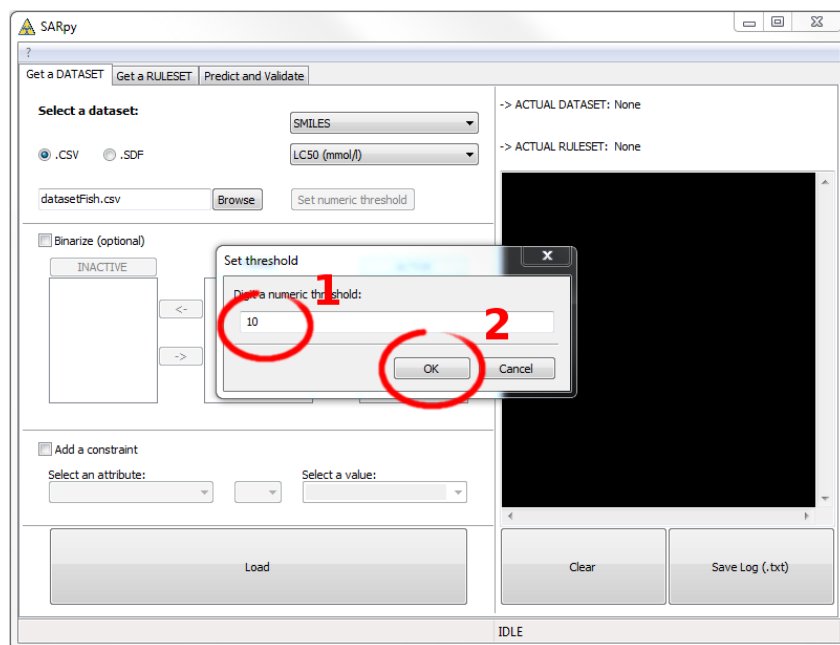


Figure 6.10: In the next popup set the numeric threshold.

Now you can, let's say, binarize your dataset; this step is totally optional. By binarize we mean splitting all the compounds in the dataset in two classes, "Active" and "Inactive", based on the mapping you specify in the three lists.

In our case, as we want the "Low" and "Moderate" class to become the "Inactive" one, and the two others to represent the "Active" class, we simply move the four labels respectively in the right or in the left list (see figure 6.11).

The same happens with a numeric threshold: you will have to decide whether the bigger or the lower values of the threshold defined above are to be set in the "Active" or "Inactive" class.

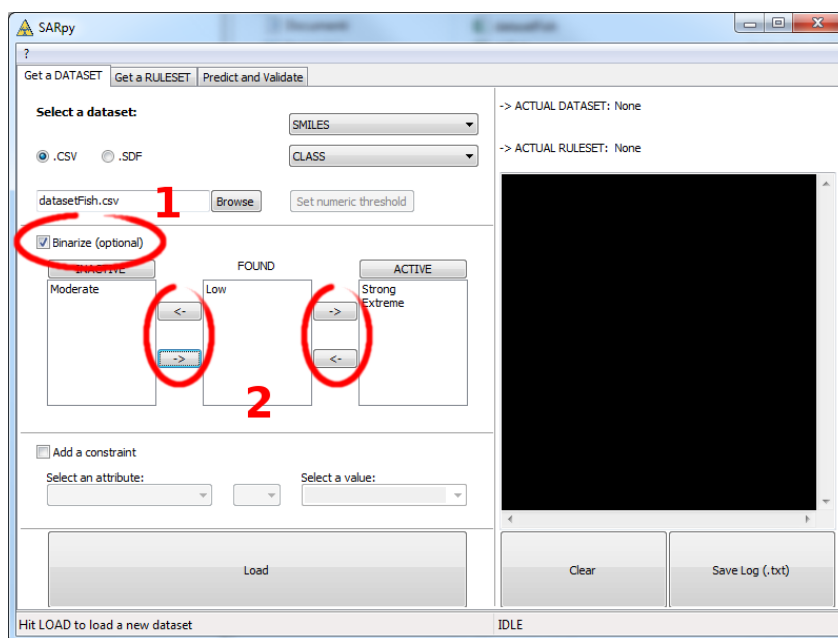


Figure 6.11: Use the arrow to move the class on the "Active" or "Inactive" side.

The last step you can perform is filtering the dataset, i.e. selecting only some specific entries and discarding all the other not meeting the property specified. In the example we want to throw away all the molecules with an ID number greater than 200 (figure 6.12).

Now that every parameter is set up, it is time to load the dataset in W-SARpy, simply clicking on the "Load" button. Read the info panel on the right to check how many molecular structures have been loaded and check on top of this if the file name is correct (figure 6.13).

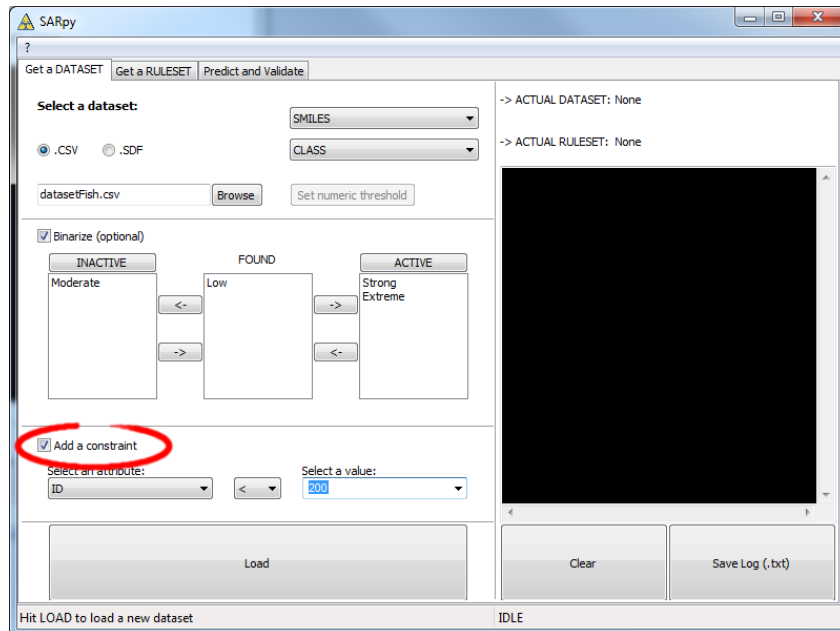


Figure 6.12: Check the “Filtering” option if you want to discard some dataset entries.

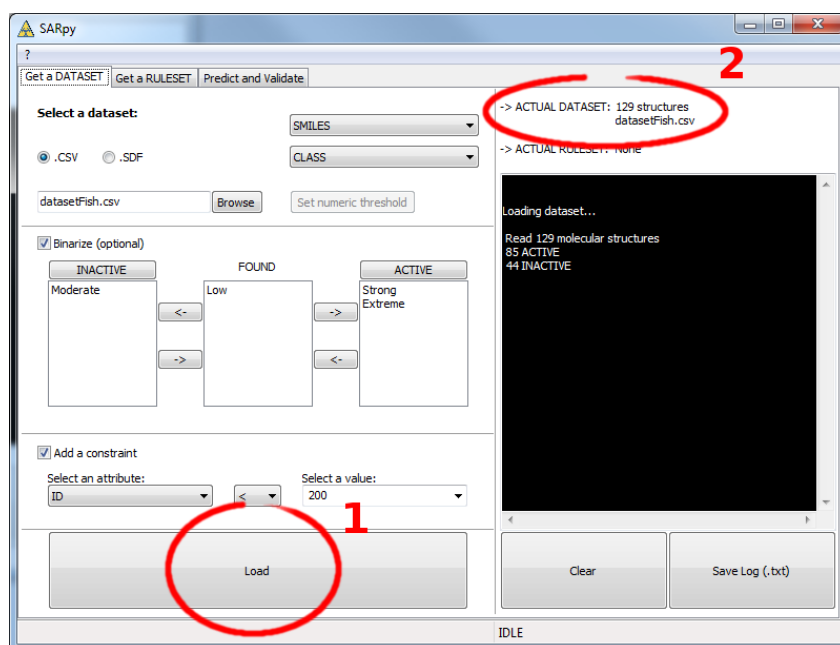


Figure 6.13: Lastly click on “Load” button to start the dataset creation process.

Loading or computing a model

The second tab is the Ruleset Tab, that will let you load or create a new model, i.e. a set of rules, from the dataset you have just loaded in the tool.

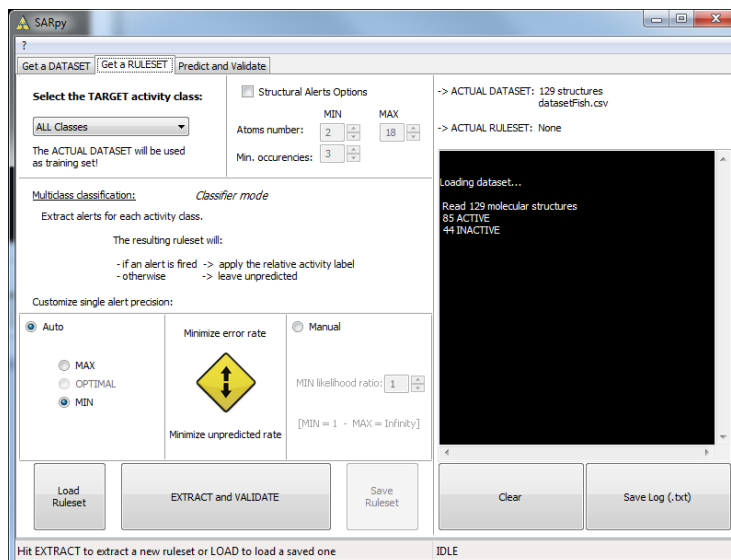


Figure 6.14: The Ruleset Tab.

Here you have just three simple parameters to setup; first of all you have to indicate which classes you want to consider for model extraction; for normal application select “All classes” as shown in figure 6.15.

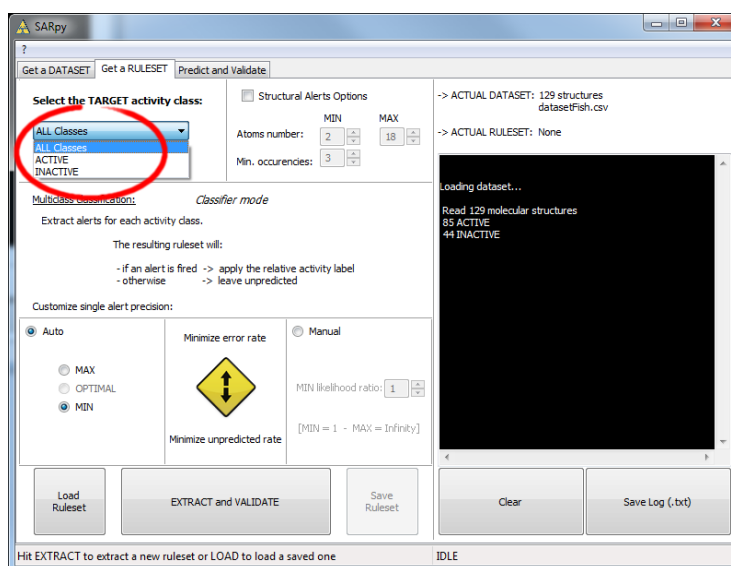


Figure 6.15: Select which class you want to consider for the ruleset extraction.

Now its time to specify the structural alerts options: you can set the minimum and maximum number of atoms in the SA and the number of occurrences needed to be considered it as significant (the higher this last number, the more precise will be the model, but will also be less restrictive).

Default values suggested are two atoms (min) and eighteen atoms (max). Changing these two numbers affect not only the number of the SAs that will be present in the model, but also has a strong influence on computation time.

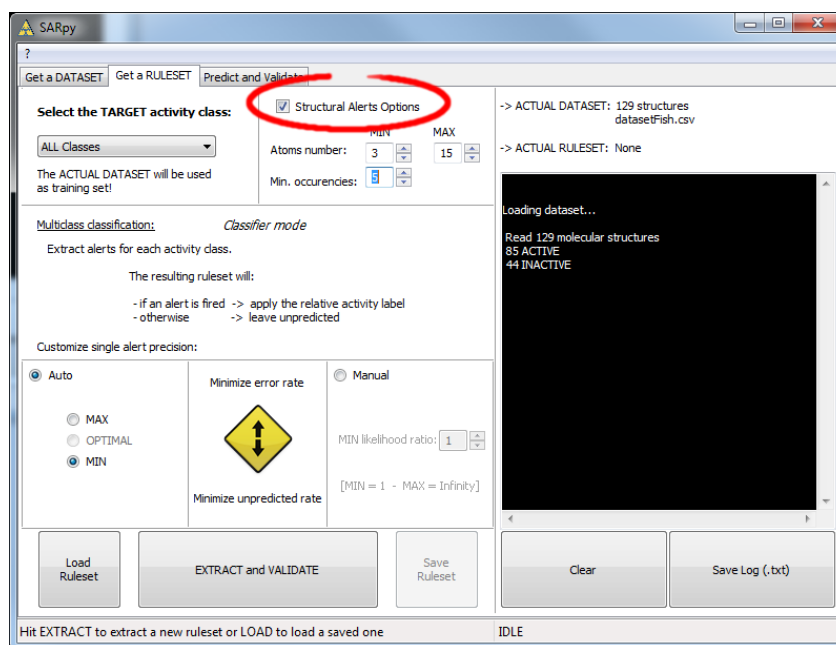


Figure 6.16: Set the values you prefer for structural alerts options

The next parameter of the model is the single alert precision, i.e. the minimum likelihood required by the model. This value affect precision and accuracy; there are two main options, “Auto” (on the left) or “Manual” (on the right). The first option let the user select among three predefined values: “Max” (that will give a more “specific” result, minimizing the error rate), “Min” (for a more “sensitive” result that minimize the unpredicted rate) and “Optimal” (a trade-off between the two). Otherwise, with the second option you can set the minimum likelihood ratio you like: increasing this parameter strengthen the precision of the model. For this example tutorial we selected “Auto-Max” mode.

To start the extraction and validation process click on the “Extract and validate” button you see in the bottom of this tab (see figure 6.17). While the process is running check the panel on the right for information about the process.

After some minutes (the computational time can vary depending on the dataset loaded and on the hardware you have), the model is created; in the info panel its accuracy and confusion matrix will be displayed. If you need you can save the resulting model, so that you can load later, maybe with a different dataset (see figure 6.18). To do this simply click on the “Save ruleset” button located on the right of the “Extract” button and select where you want to store the file, as shown in figure 6.19.

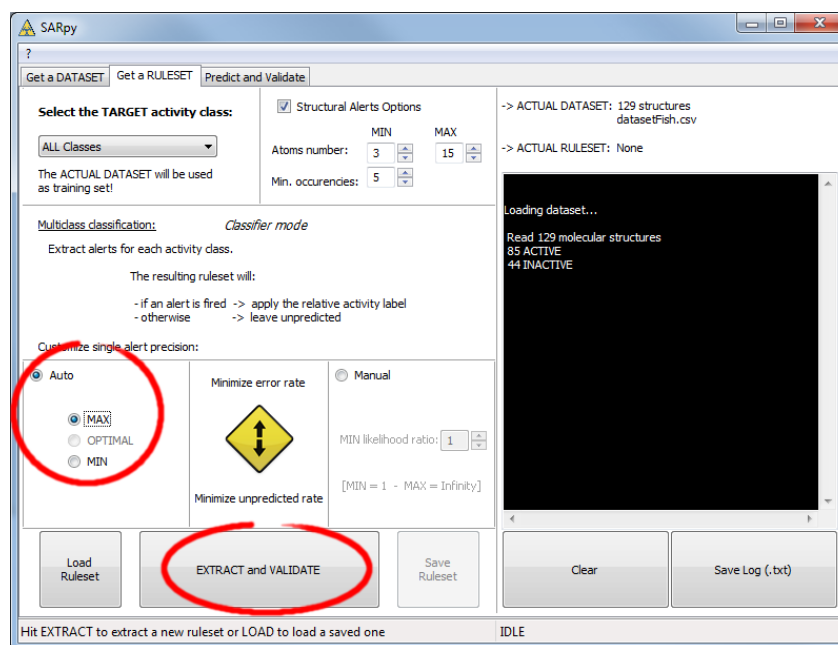


Figure 6.17: Set the single alert precision you prefer; a higher LR will result in a higher precision of the model.

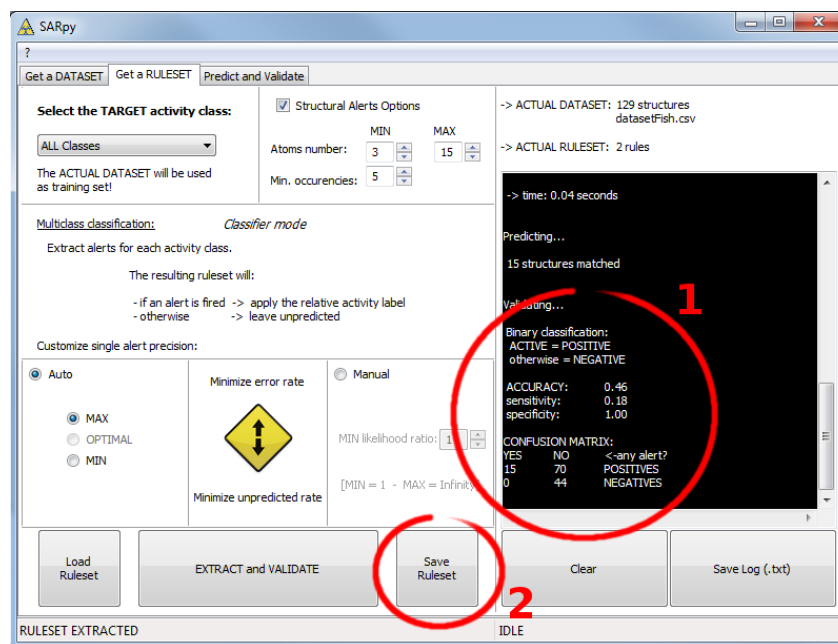


Figure 6.18: When the process is over the info panel will show some information about the model.

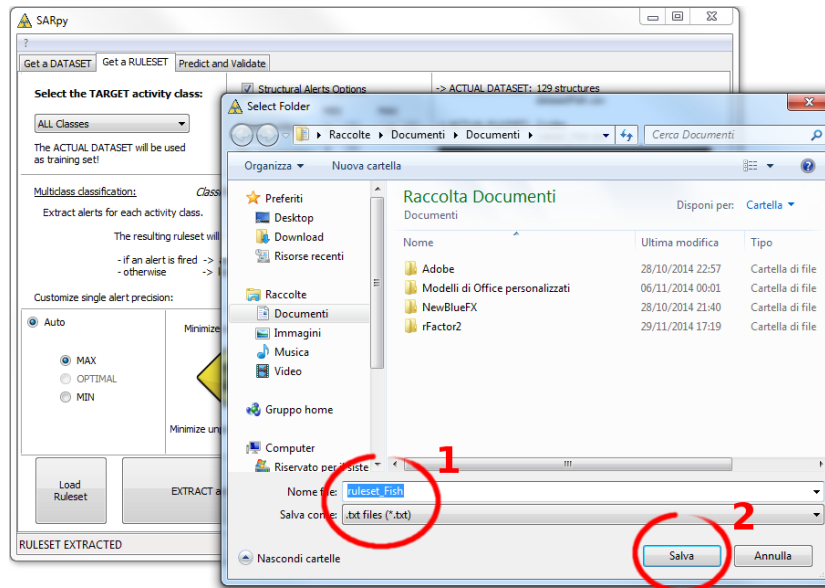


Figure 6.19: When saving the ruleset make sure the .txt format is selected.

A plain text file will be created. Its structure is quite simple and is shown in figure 6.20: each row is a rule, and contains the fragment SMART, the training class (as “Active” or “Inactive”) and the normalized likelihood ratio.

```

ruleset_Fish - Blocco note
File  Modifica  Formato  Visualizza  ?
SMARTS Target Training LR
C(C1)(C1) ACTIVE inf
c1cccc1C(=O)OCC INACTIVE inf
CC(C)CC(O) ACTIVE inf
C1CCC ACTIVE inf
c1cc(ccc1N)C(=O) ACTIVE 3.54
c1c(C)cc(N(=O)=O)cc1 ACTIVE inf
COCCOC ACTIVE inf
CCCCC(=O)C INACTIVE 5.93
CCCCCCCC ACTIVE 1.62
Cc1cc(C1)ccc1 INACTIVE 4.95
CCC#C ACTIVE 3.54
c1ccc(Oc2cccc2)cc1 ACTIVE 4.04
c1cc(C1)ccc10 INACTIVE 4.62
c1c(C1)cccc1C1 ACTIVE 1.01
C=CC(=O)OCC ACTIVE 4.04
NC(=O)C ACTIVE 3.54
CCC(C)CC INACTIVE 1.32
C1C ACTIVE 6.07

```

Figure 6.20: An example of the ruleset file as created by W-SARpy.

Predicting and validating

Once you have correctly loaded a dataset and a ruleset, you can switch to the last tab, the “Predict and validate” tab; here you can predict the activity of other unseen compound, i.e. apply the ruleset on the dataset, and also validate the model, i.e. compute the prediction error and the confusion matrix. This tab is shown in figure 6.21.

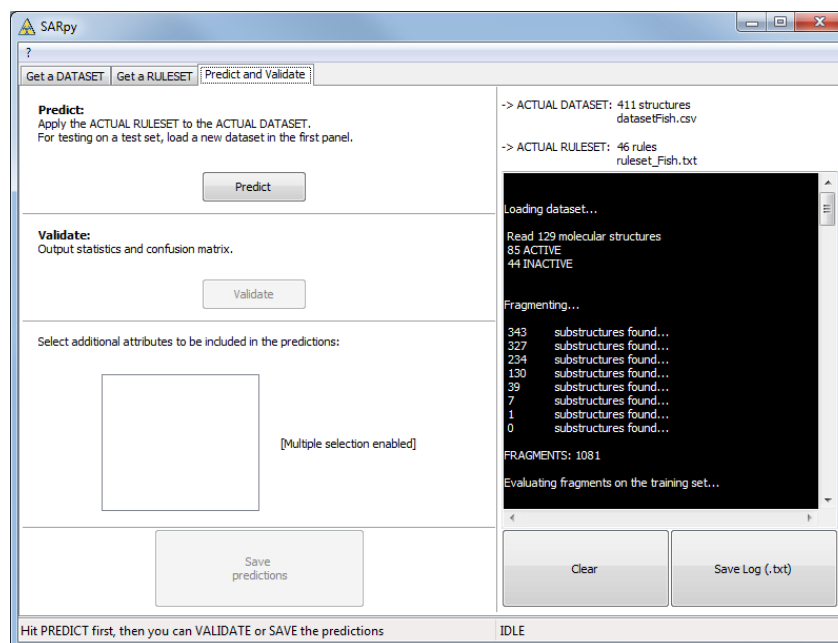


Figure 6.21: The “Predict and validate” tab.

To predict the property of each compound in the selected dataset simply click on the first button, “Predict”; the info panel will show you how many structures have been matched (figure 6.22).

If you need it, you can save the prediction results as text file; just click on the “Save predictions” button, select the folder and the file name and make sure the selected format is .txt. Optionally you can select some attributes of your dataset to be added to the prediction output. To do this, select those you want from the list (see figure 6.23); multiple selection is performed by pressing and holding the shift key while clicking on attributes. These attributes are in addition to those saved by default.

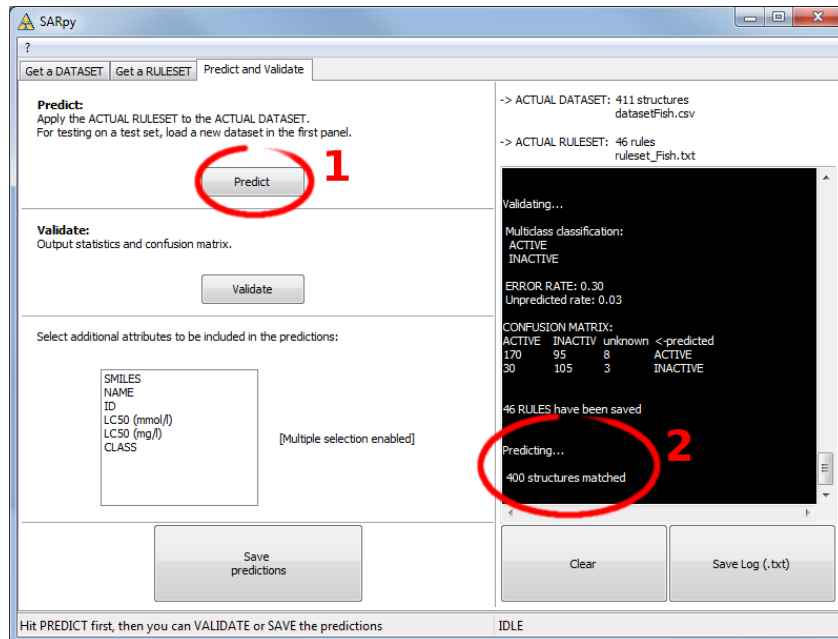


Figure 6.22: Click on “Predict” to run the prediction process.

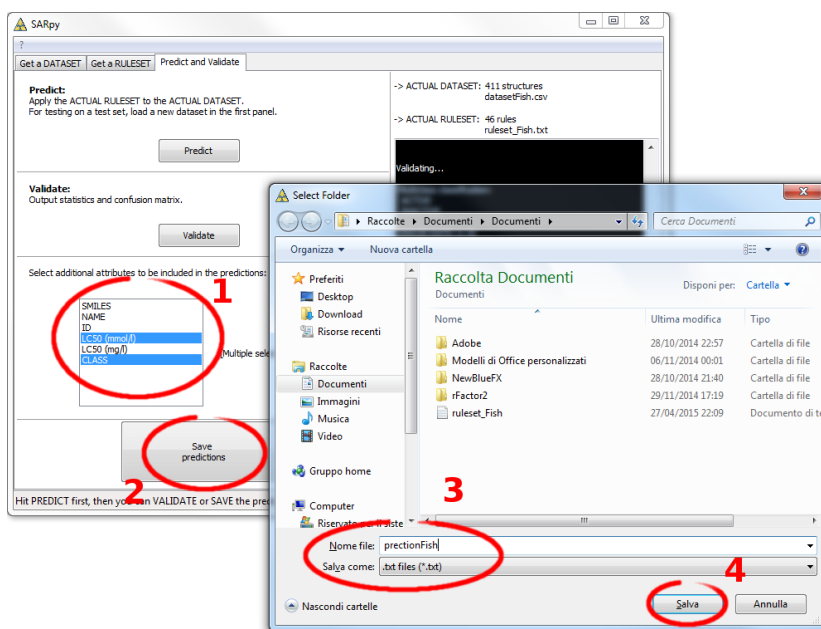


Figure 6.23: How to save the prediction result as text file.

The file produced in this example is shown in figure 6.24; each compound is listed in a row, with its SMILE, the predicted class (Active or Inactive), the likelihood ratio and the SMARTS, also with the LC50 value and the original class of the compound.

SMILES	Prediction	Training LR	SMARTS	LC50 (nmol/l)	CLASS
<chem>C(C1)(C1)C(=O)N</chem>	ACTIVE	inf	<chem>C(C1)(C1)</chem>	1.8834011	Extreme
<chem>n1c(O)c(C#N)c(C)cc1c</chem>	ACTIVE	2.53	<chem>n1c(C)cccc1</chem>	1.0595937	Extreme
<chem>c1ccccc1Ss1cccc1</chem>	INACTIVE	1.07	<chem>c1ccccc1</chem>	0.0005038	Extreme
<chem>oc1c(O)c(C1)c(C1)c(C1)c1c1</chem>	INACTIVE	1.77	<chem>c1cc(O)c1</chem>	0.0051232	Extreme
<chem>O=C(N)OCC</chem>	INACTIVE	1.27	<chem>C(=O)OC</chem>	58.8169267	Extreme
<chem>CCCCOC(=O)CCC(=O)OCCCC</chem>	ACTIVE	1.77	<chem>CCOCCCC</chem>	0.0193610	Strong
<chem>CCCCOCCC</chem>	ACTIVE	1.77	<chem>CCOCCCC</chem>	0.2480227	Strong
<chem>OC(=O)C1(C)CCCC2(C)c3ccc(C(C)C)cc3CCC12</chem>	ACTIVE	inf	<chem>CC(C)CC(O)</chem>	0.0069897	Extreme
<chem>c1(O)cccc1C(=O)N</chem>	ACTIVE	1.64	<chem>c1(O)cccc1C(=O)</chem>	0.7364737	Extreme
<chem>CNC(=O)oc1cccc2cccc12</chem>	ACTIVE	1.26	<chem>CNC(=O)</chem>	0.0443771	Extreme
<chem>oc1c(N(=O)=O)c(C)Ncc1</chem>	ACTIVE	1.52	<chem>c1cc(O)cccc1N</chem>	0.2348667	Extreme
<chem>c1ccoc1</chem>	None	0.8961363	<chem>Strong</chem>		
<chem>CCCCCCCCC(=O)C</chem>	INACTIVE	5.93	<chem>CCCC(=O)C</chem>	0.0064019	Extreme
<chem>CCOP(=S)(OC)SC5C(C)C(C)C</chem>	ACTIVE	2.02	<chem>C5C</chem>	0.0000461	Extreme
<chem>CCOC(=O)c(C1)C(=O)OCC</chem>	ACTIVE	6.07	<chem>c1c</chem>	0.0048816	Extreme
<chem>CCCCC1OC(=O)CC1</chem>	ACTIVE	1.77	<chem>CCOCCCC</chem>	0.1057269	Strong
<chem>oc1c(C(C)(C)C)cc(C(C)C)C(C)C(C)C</chem>	INACTIVE	1.38	<chem>c1ccc(C(C)C)cc1</chem>	0.0002321	Moderate
<chem>c1c(=C(C)C)C=C2CC3C(C)C(=O)O)CCCC3(C)C2C1</chem>	ACTIVE	inf	<chem>CC(C)CC(O)</chem>	0.0056536	Strong
<chem>c1(C=O)ccc(Oc2cccc2)cc1</chem>	ACTIVE	4.04	<chem>c1ccc(Oc2cccc2)cc1</chem>	0.0232065	Strong
<chem>oc1c(C1)cc(C1)c1c1</chem>	INACTIVE	4.62	<chem>c1cc(C1)ccc10</chem>	0.0463915	Strong
<chem>CCCCN</chem>	ACTIVE	1.61	<chem>C(N)C</chem>	3.6642056	Strong

Figure 6.24: An example file containing the saved predictions.

The last tool you may want to use is the validation, useful to compute the error rate and the confusion matrix of the model when applied to the loaded dataset (figure 6.25).

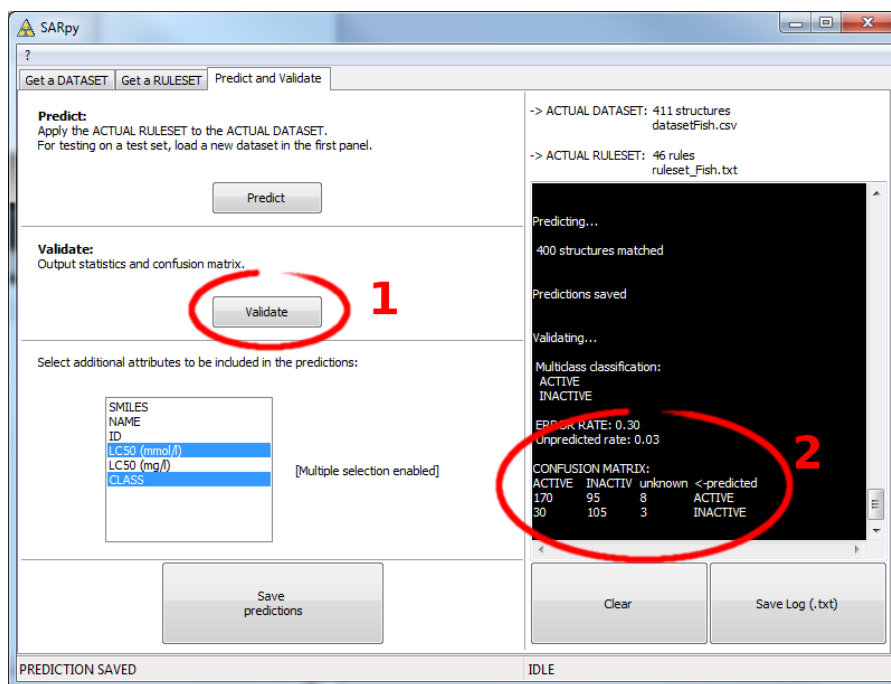


Figure 6.25: To validate the model click on the “Validate” button and read the result in the info panel.

6.3 S-SARpy

When your .csv file is ready you can start S-SARpy. Open a terminal and start a python interactive shell; if you prefer you can also collect all the commands we will present you in a python script.

In the python shell send the following commands:

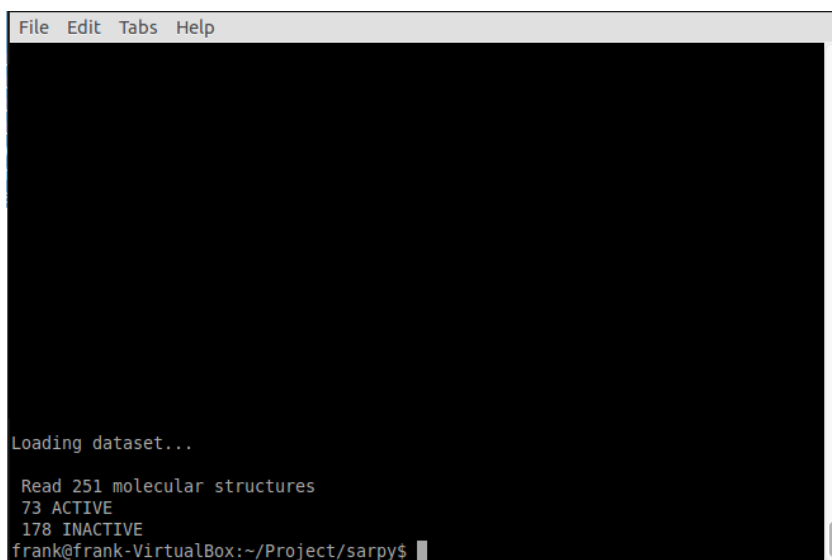
```
1 import SARpy
2 import operator
```

Now you are ready to call all the function defined in the S-SARpy tool.

Loading the dataset

The first step is to load the .csv file you have created. During this phase you can also control binarization and filtering: for binarization you just have to create all the filtering rules you need and collect them in a dictionary; for filtering simply create the rule and specify it when calling the “loadDataset” function. For this example we have created four rules for binarization that map the two classes “Low” and “Moderate” and the two “Strong” and “Extreme” respectively in the “Inactive” and “Active” class. No filtering is performed. The output is shown in figure 6.26.

```
1 f1 =SARpy.Filter('CLASS','Low',operator.eq)
2 f2 =SARpy.Filter('CLASS','Moderate',operator.eq)
3 f3 =SARpy.Filter('CLASS','Strong',operator.eq)
4 f4 =SARpy.Filter('CLASS','Extreme',operator.eq)
5 dict = {'INACTIVE':f1,'INACTIVE':f2,'ACTIVE':f3,'ACTIVE':f4}
6 dataset = SARpy.loadDataset("ds.csv", "csv", dict, "SMILES")
```



```
File Edit Tabs Help

Loading dataset...

Read 251 molecular structures
73 ACTIVE
178 INACTIVE
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 6.26: The output of the “loadDataset” function shows you the number of structures loaded

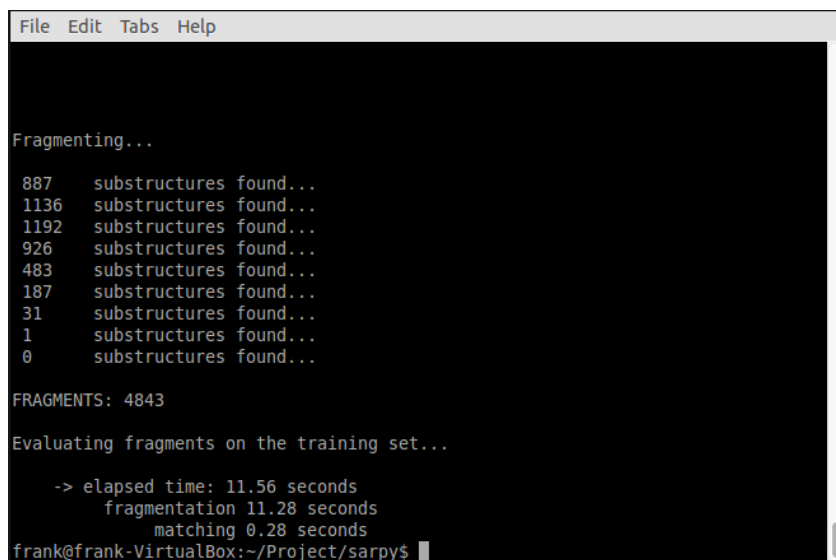
Computing a model

Now you can compute a new model; the first function to be used is “fragmentize”, which extract all the structural alerts from the dataset. You have to specify the number of minimum and maximum number of atoms in the SA (no default values are set); we selected 2 and 18 respectively. The output is show in figure 6.27.

Then you have to extract the rules; use the “extract” function and specify the dataset and the parameters of the model; in order these are: minimum number of hits (default value is 3), minimum likelihood ratio (default value is 1), minimum precision (no default value) and the target activity class (default value is ‘None’). The process will last some minutes, depending on the selected parameters and the hardware you have.

Optionally you can save the extracted rules in a plain txt file (see line number 3); the generated file is shown in figure 6.29.

```
1 SARpy.fragmentize(dataset,2,18)
2 rules = SARpy.extract(dataset, 3,1,None)
3 SARpy.saveSmarts(rules, "ruleset.txt")
```



```
File Edit Tabs Help

Fragmenting...

887  substructures found...
1136 substructures found...
1192 substructures found...
926  substructures found...
483  substructures found...
187  substructures found...
31   substructures found...
1    substructures found...
0    substructures found...

FRAGMENTS: 4843

Evaluating fragments on the training set...

-> elapsed time: 11.56 seconds
    fragmentation 11.28 seconds
    matching 0.28 seconds
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 6.27: Read the output of the fragmentation to check if the process ended correctly.

```

File Edit Tabs Help

Extracting rules...

2427 ACTIVE substructures
233 of which are potential alerts

2995 INACTIVE substructures
189 of which are potential alerts

Extracted:
12   ACTIVE
15   INACTIVE

RULES: 27

-> time: 0.20 seconds
frank@frank-VirtualBox:~/Project/sarpy$

```

Figure 6.28: Some details of the model will be shown as soon as the computation end.

SMARTS	Target	Training	LR
CCCCCCCCC	ACTIVE	inf	
c1cc(Cl)ccc10	ACTIVE	inf	
CCCCOCc1ccccc1	ACTIVE	inf	
0c1c(C(C)C)cccc1	ACTIVE	inf	
c1(C=O)ccc(O)cc1	INACTIVE		4.10
C(=O)CCCC	INACTIVE		1.85
c1cc(C#N)ccc1	INACTIVE		inf
Cc1c(N)cccc1	INACTIVE		inf
CN(C)C	INACTIVE	inf	
CN(C)c1ccccc1	INACTIVE		inf
c1c(F)cccc1	INACTIVE		inf
c1ccccc1C(=O)OC	INACTIVE		1.23

Figure 6.29: An example of the ruleset file as created by S-SARpy.

Predicting and validating

Once you have correctly loaded a dataset and a ruleset you can predict the activity of other unseen compounds (i.e. apply the ruleset on the dataset) and also validate the model (i.e. compute the prediction error and the confusion matrix).

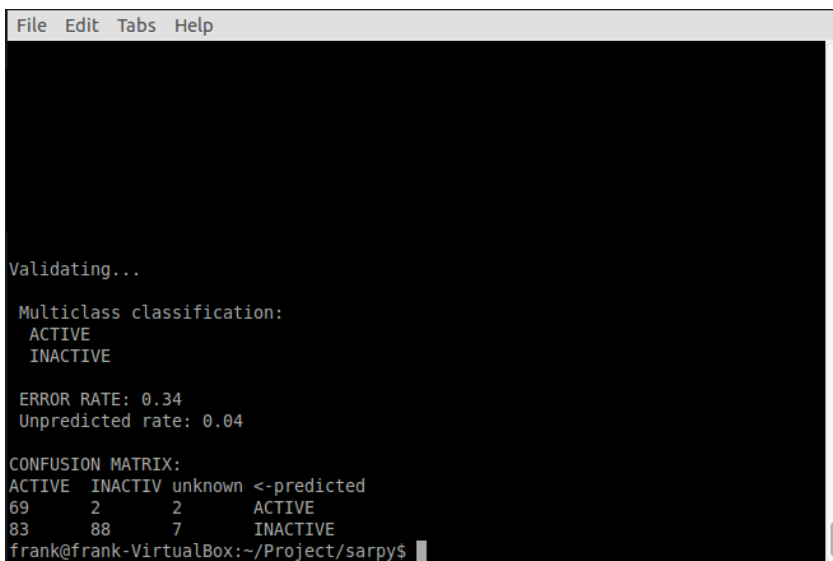
You have to simply call the “predict” function, specifying the ruleset and the dataset; the result is shown in figure 6.30. After that you can optionally save the predictions in a plain .txt file for further analysis and also validate the generated model (see figure ?? for details).

```
1 pred = SARpy.predict(rules, dataset)
2 SARpy.savePredictions(dataset, "prediction.txt")
3 SARpy.validate(dataset)
```

A terminal window with a black background and white text. The text shows the prediction process: "Predicting...", "242 structures matched", and the prompt "frank@frank-VirtualBox:~/Project/sarpy\$".

```
Predicting...
242 structures matched
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 6.30: The number of predicted structure is shown as the process is over.

A terminal window with a black background and white text. The text shows the validation process: "Validating...", "Multiclass classification:", "ACTIVE", "INACTIVE", "ERROR RATE: 0.34", "Unpredicted rate: 0.04", and a "CONFUSION MATRIX:" table. The prompt is "frank@frank-VirtualBox:~/Project/sarpy\$".

```
Validating...

Multiclass classification:
ACTIVE
INACTIVE

ERROR RATE: 0.34
Unpredicted rate: 0.04

CONFUSION MATRIX:
ACTIVE  INACTIV  unknown  <-predicted
69      2         2         ACTIVE
83      88        7         INACTIVE
frank@frank-VirtualBox:~/Project/sarpy$
```

Figure 6.31: The validation process will inform you about the performance of the model.

```
SMILES,Prediction,Training LR,SMARTS,CLASS,ID
c1c(C(=O)CBr)c(OC)ccc1OC,ACTIVE,1.72,0c1cccc1,Extreme,462
CC1(C)Oc2c(OC(=O)NC)cccc2C1,ACTIVE,1.72,0c1cccc1,Extreme,467
Oc1c(Cl)cc(Cl)c(Cl)c1Cc1c(Cl)c(Cl)cc(Cl)c10,ACTIVE,inf,c1cc(Cl)ccc10,Extreme,47
Nc1ccc(F)cc1,INACTIVE,inf,c1c(F)cccc1,Moderate,337
CCOP(=S)(OCC)Oc1nc(Cl)c(Cl)cc1Cl,ACTIVE,9.75,P(=S)(OCC)0,Extreme,513
C=C(CCl)CCL,ACTIVE,3.55,C=C,Extreme,477
c1c(Cl)c(O)c(Cl)cc1C#N,INACTIVE,inf,c1cc(C#N)ccc1,Moderate,478
n1c(Cl)c(Cl)c(Cl)c(Cl)c1Cl,None,,Extreme,490
c1(C=O)cc(C(F)(F)F)ccc1,ACTIVE,1.12,c1cccc1,Extreme,342
Nc1c(Cl)c(Cl)cc(Cl)c1Cl,ACTIVE,5.69,c1c(Cl)cccc1Cl,Extreme,519
```

Figure 6.32: The file “prediction.txt” generated by S-SARpy.

Bibliography

- [1] Ferrari, T., Gini, G., Bakhtyari, N. G., and Benfenati, E. Mining structural alerts from smiles: a new way to derive structure-activity relationships. *Proc. IEEE SSCI 2011: Symposium Series on Computational Intelligence* (2011), 120–127.
- [2] Gini, G., Ferrari, T., Cattaneo, D., Bakhtyari, N. G., Benfenati, E., and Manganaro, A. Automatic knowledge extraction from chemical structures: the case of mutagenicity prediction. *SAR and QSAR in Environmental Research* 24, 5 (2013), 365–383.
- [3] Lombardo, A., Pizzo, F., Benfenati, E., Manganaro, A., Ferrari, T., and Gini, G. A new in silico classification model for ready biodegradability, based on molecular fragments. *Chemosphere* 108 (2014), 10–16.
- [4] Russom, C. L., Bradbury, S. P., Hammermeister, D. E., and Drummond, R. A. Predicting modes of toxic action from chemical structure: acute toxicity in the fathead minnow (*pimephales promelas*). *Environmental Toxicology Chemistry* 16 (1997), 948–967.