Deliverable D5.1

Report on the Initial Prototype for Validation and Reporting

Grant Agreement	Health-F5-2008-200787
Acronym	OpenTox
Name	An Open Source Predictive Toxicology Framework
Coordinator	Douglas Connect

Contract No.	Health-F5-2008-200787	
Document Type:	Deliverable Report	
WP/Task:	WP5	
Name	Report on Initial Prototype of Validation Routines	
Document ID:	OpenTox Deliverable Report 1 WP5	
Date:	March 8, 2010	
Status:	Final Version	
Organisation:	Albert-Ludwigs Universität, Freiburg	
Contributors	Andreas Karwath	ALU-FR
	Martin Gütlein	ALU-FR
	Barry Hardy (review)	DC
	Nicki Douglas	DC

Distribution:	Public

Purpose of Document:	To document results for this deliverable

Document History:	1 – Initial draft prepared on Feb 12, 2010
	2 – Some minor changes, report examples added to appendix, Feb. 19, 2010
	3 – Added validation objects, and future work, Feb. 27, 2010
	4 – Final approved version, Feb 28, 2010
	5 – Updated version, March 8, 2010

Table of Contents

Summary

We followed the OECD Principle[1] for the Validation of (Q)SAR "Appropriate Measures of Goodness-of-Fit" as a strong guidance, while developing the Open validation web service. To this end, the main aims of the validation web service are to provide the means to critically evaluate predictive toxicology models and algorithms on a standardized platform, as well as to provide a common reporting standard for these comparisons. Such validation capabilities are an important reporting requirement for REACH-based evaluation and acceptance of *in silico*-based (Q)SAR models.

The OpenTox validation and reporting routines are implemented following the RESTful Web Service architecture, comply with the current OpenTox Application Programming Interface (API 1.1) and are available as Open Source programs. The OpenTox routines provide the following functionality:

- State-of-the-art validation procedures such as training-test-set validation and cross-validation can be performed[2], including the calculation of various statistical figures that describe the quality of the evaluated model or algorithm;
- The validation was applied to partner modelling web services (classification algorithm Lazar and feature generation service Fminer)[3];
- OpenTox Validation reports can be generated automatically, to present the validation results in a user-friendly way, including charts;
- The validation web service was integrated into the OpenTox-based ToxCreate application[4] to validate predictive models created by ToxCreate users.

[1] ecb.jrc.ec.europa.eu/qsar/background/index.php?c=OECD

[2] opentox.informatik.uni-freiburg.de

[3] github.com/helma/opentox-algorithm/

[4] www.opentox.org/toxicity-prediction/

1.Introduction

Numerous diverse approaches for predicting toxicity via (Q)SAR have been proposed in the literature. Many of these approaches are included within this project in the algorithm Work Package (WP). To be able to reliably compare models as well as algorithms, it is however necessary to standardize the comparison routines. This has been the main aim in the first part of this model validation WP: to provide the means to evaluate models and algorithms on a standardized platform. Furthermore, it is also necessary to provide a common reporting standard for these comparisons. For the integration of these validation and reporting tools, a common collaboration framework based on the RESTful web service architecture was defined for all services in the proposed OpenTox Framework. This allows seamless integration of novel algorithms and other validation services, based on well-established protocols.

The prototypes for each service is described using a tabular format giving a brief overview of the service, including partners and contact persons, as well as providing links to the resources. This overview is followed by technical information about required and optional parameters, status codes and their meaning, output of the service, the programming language employed, and an example.

2.Implementation Principles

The validation and reporting framework was implemented according to the following principles:

2.1 Open Source programming tools

As the open source philosophy is inherently important for this project, all tools developed are openly available via public repositories. The main language used in the development of the validation prototype is ruby[5]. Other applications used are also open source, and include e.g. Apache[6].

[5] www.ruby-lang.org

[6] httpd.apache.org

2.2 RESTful Web Service Architecture

All current OpenTox web services adhere to the Representational State Transfer (REST) Web service architecture for sharing data and functionality among loosely-coupled, heterogeneous systems. The REST web service architecture has a number of desired advantages when compared to other architectures:

1. It is lightweight, as only some additional xml mark-up is required;
2. The produced results are human-readable, i.e. the resources are uniquely identified by URIs and described by representations;
3. RESTful web services are typically stateless[7];
4. The produced web services have a uniform interface (the only allowed operations are the HTTP operations);
5. Components manipulate resources by exchanging representations of the resources.

All validation and reporting resources have representations providing information about the type of validation performed, the original data set used, the random seed used for splitting (in the case of a k-fold-cross validation), or which algorithm was used for the validation. As the exchange format, the Resource Description Framework (RDF) representation[8], in particular the XML-formated version, was chosen.

1. RDF is a W3C recommendation: RDF-related representations such as rdf/xml and rdf/turtle are w3c recommendations so they constitute a standard model for data exchange;
2. RDF is part of Semantic Web Policy: RDF as a representation for a self-contained description of web resources contributes to the evolution of the Semantic Web; a web where all machines can “understand” each other;
3. RDF is designed to be machine-readable: While a human user can read an RDF document, it is unlikely they will to be able to understand it (at least not easily). RDF is intended to be understood by computers, not people.

Some services support additional representations like YAML[9] (YAML Ain't Markup Language).

---

[7] www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[8] www.w3.org/RDF/

[9] www.yaml.org/

2.3 OpenTox Validation and Reporting Programming Interfaces (APIs)

Validation and Reporting APIs are included in the OpenTox API ensuring the seamless interaction between all OpenTox components with regards to validation and reporting needs. The current OpenTox API version is API 1.1 (www.opentox.org/dev/apis/api-1.1). Each validation and reporting component is, according to the design specifications above, a resource. Each validation resource for example, contains information about the dataset and the model, so the underlying procedures can be invoked.

We use the notation '/resource' to denote the class of URIs someDomain.com/resource where someDomain.com can be the domain name of any OpenTox server (such as opentox.informatik.uni-freiburg.de). We use sub-URIs to distinguish different web services: e.g. opentox.informatik.uni-freiburg.de/validation for the validation web service. All validation resources share this prefix. For example opentox.informatik.uni-freiburg.de/validation/1 is the result of a plain test-set validation with ID 1, opentox.informatik.uni-freiburg.de/validation/2 is the resource of a cross-validation with ID 2.

The **validation API** consists of a number of operations that are described in the following section. Each operation uses one of the following HTTP methods: GET, PUT, POST, or DELETE[10]:

1. GET /: Returns a list of all validations on the server. Currently supported return formats are: text/uri-list

2. GET / {id}: Returns a representation of a specific validation object, identified by its id. The requested MIME type should be set in the requests "Accept" header, e.g. 'curl -X GET -H "Accept:application/xml" http://{server}/validation/{id}'. The supported MIME types are: application/rdf+xml and text/x-yaml (default).

3. POST /: Validates a model on a test dataset. Two different versions are provided:

   - The first version validates an already existing model. It requires the parameters 'model_uri', 'test_dataset_uri' and 'prediction_feature'. The result of such an

[10] www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

operation is a task URI, as the validation process will be executed asynchronously. As soon as the validation is finished the task will provide the validation URI.

- The second version creates a model with a training dataset first, then validates the model on a test dataset. The model building step is automatically performed if the given parameters are 'algorithm_uri', 'prediction_feature', 'algorithm_params', 'training_dataset_uri', and 'test_dataset_uri'. The result is again a task, that links to the validation resource as soon as it is finished. In case the user wants to compare a model to models trained on a y-scrambled dataset (www.qsarworld.com/qsar-statistics-y-scrambling.php), the optional 'y_scramble' (default=false) and 'y_scramble_seed' (default=1) can be used.

4. POST /training_test_split is similar to the operation just described, but splits one single dataset into training and test sets and then performs model construction on one part of the dataset while evaluating the learned model on the remainder of the original dataset. The required parameters are as follows: 'algorithm_uri', 'prediction_feature', 'algorithm_params', 'dataset_uri', 'split_ratio' (default=0.66), 'random_seed' (default=1), and the optional 'y_scramble' and 'y_scramble_seed'. The result is again a validation URI or a task URI.

5. DELETE /{id} deletes a specified validation object.

Detailed information about the validation API can be found at www.opentox.org/dev/apis/api-1.1/Validation.

The same design concepts were used in the construction of the Cross-Validation API. A cross-validation component performs k single validations using a standard k-fold cross-validation.

1. GET /crossvalidation: Returns a list of all cross-validations on the server. Currently supported return formats are: text/uri-list

2. GET /crossvalidation/{id}: Returns a representation of a specific cross-validation object, identified by its id. The requested MIME type should be set in the requests "Accept" header. The supported MIME types are: application/rdf+xml and text/x-yaml (default).

3. GET /crossvalidation/{id}/validations: Returns a list of validation objects associated

with a specific cross-validation object, identified by its id. The result is a list of validation URIs.

4. POST /crossvalidation: A POST operation on a validation activates the application of a k-fold cross-validation routine, given a given dataset and an algorithm. The result of such an operation is a cross-validation URI or a task URI. The following parameters have to be submitted: ‘algorithnm_uri’, ’prediction_feature’, ’algorithm_params’ (default=""), ’num_folds’ (default=10), ‘random_seed’ (default=1), ‘stratified’ (default=true), ‘y_scramble’ (default=false), and ‘y_scramble_seed’ (default=1).

5. POST /crossvalidation/loo: This performs a leave-one-out cross-validation, resulting in a cross-validation URI or a task URI. In contrast to the normal k-fold cross-validation the parameters for splitting the dataset are not required. The rest of the parameters are the same.

6. DELETE /crossvalidation/{id} deletes a specified validation object.

The same architectural concept was applied to the construction of the **reporting API**, which provide reporting capabilities for all validation objects.

1. GET /report: Retrieves a list of reports types stored on the server. The available report types are described in detail below.

2. GET /report/{report-type}: Retrieves a list of all reports for the particular report type. The result is a list of available reports as URI.

3. GET /report/{report-type}/{id} : Retrieves a specific report for the particular report type in one of the following output formats: XML, PDF, HTML, or RTF. The accept-header is used to differentiate between different output formats.

4. POST /report/{report-type}: Creates a report for the specific report types. The parameters, as well as the results, are report-type specific and are detailed below:

   a. POST /report/toxpredict: This creates a ToxPredict report. ToxPredict is one of the initially selected OpenTox prototype use cases where the user requires a number of models predicting toxicity for one or a library of compounds. The required parameters are a ‘list of validation URIs’ representing the outcome of the employed models. The result is a report URI or task URI.

b. POST /report/validation: This operation creates a single validation report for one model applied to one dataset. The required parameter is a 'validation URI'. The result is a report URI or task URI.

c. POST /report/crossvalidation: This creates a cross-validation report for one algorithm applied to one dataset, split into k folds. The required parameter is a 'crossvalidation URI'. The result is a report URI or task URI.

d. POST /report/algorithm_comparison: This creates a report to compare the performance of different algorithms when applied to the same datasets and cross-validation splits. The report shows which algorithm performs better on the dataset, and compares its overall performance. Again 'validation URI's' or 'cross-validation's' are required as input parameters, with the prerequisite that there is one validation for each algorithm and cross-validation/test dataset provided.

e. POST /report/model_comparison: Creates report for comparing different models, i.e. multiple models applied to one dataset. The required parameter is a 'list of validation URIs'

f. POST /report/qmrf: Creates a QMRF report (see: ecb.jrc.ec.europa.eu/qsar/qsar-tools/index.php?c=QRF). As this kind of reporting format can only partially be filled automatically, it is intended to produce a report in XML or RTF format, which can later be filled either via the QPRF Editor[11] or within a word processing program like OpenOffice Writer[12]. The required parameters are a list of cross-validation URIs and/or validation URIs of the same model. The result is a report URI or task URI.

g. POST /report/qprf: Creates a QPRF report. As with the QMRF reporting format, this can only partially be filled using automated methods. Therefore, we plan to partially fill the report, but include all relevant fields for manual editing.

5. DELETE /report/{report-type}/{id}: Deletes a specified report based on the report

[11] http://ambit.sourceforge.net/qmrf/jws/qmrfeditor.jnlp

[12] www.openoffice.org

type and id.

More information about the validation and reporting API is available at the address www.opentox.org/dev/apis/api-1.1/Validation.

3. Validation and reporting prototype presentation

The web service objects and routines in the validation and reporting prototype are described in section 4 below. The table-based description of these web service routines is structured as follows: a general description of the service is given, followed by input parameters and output values, relevant status codes, implementation information, and finally an example. Here we have solely concentrated on PUT operations, as the GET and DELETE operations just return lists of URIs or delete resources.

3.1 General Information about the service

The first table section provides a description of the service, accompanied by the URI of the resource and a reference to the current API. Other fields of this table indicate the partner who has been responsible for the development of the service, and a contact person who can provide more information about the implementation. The last comments field can be used for any further comment on the service including reviews.

3.2 Request/Response Information

As described in the API section of this document, each service requires a number of input parameters. In the second table section, these parameters are listed. All parameters having default values are optional, the ones without are mandatory. In a k-fold cross-validation, the random seed to split is default 1, therefore optional while the algorithm URI is mandatory. The table also lists the result of each operation.

3.3 Status Codes

This part of the table presents and describes the list of standard response codes that can be produced by each web service, according to the APIs. The codes help identify the cause of the problem when the service is not working properly. The term HTTP status code is actually the common term for the HTTP status line that includes both the HTTP status code and the HTTP reason phrase. For example, the HTTP status line 500: Internal Server Error is made up of the HTTP status code of 500 and the HTTP reason phrase of Internal Server Error.

3.4 Implementation Information

The fourth table section provides technical details about the prototype implementation, such as the type of HTTP method used to execute the service, the programming languages employed, and the open source libraries used (if applicable) which were integrated within the service.

3.5 Examples

In the last table section, an example for each web service is provided. The example is based on the curl[13] commands and can be easily invoked from the command line and can be used by technical reviewers to test and evaluate the performance of each service.

4. Prototype documentation

4.1 Validation Result

Each validation routine evaluates a model on a test dataset and returns a validation object as result. The validation object contains various quality figures as well as Meta information (see table below). An even more detailed definition of the object can be found in the OpenTox Ontology[14].

Type	Value	Description
Meta information	training_dataset_uri	
	test_dataset_uri	
	prediction_dataset_uri	Dataset that contains model predictions
	prediction_feature	Predicted feature
	uri	URI of the validation object itself
	model_uri	
	real_runtime	Time needed for validation
	id	
	created_at	
General validation information	num_instances	Number of instances in the test dataset
	num_without_class	Number of instances with missing class values
	percent_without_class	Percent of instances with missing class values
	num_unpredicted	
	percent_unpredicted	

[13] curl.haxx.se

[14] www.opentox.org/data/documents/development/RDF%20files/OpenToxOntology/view

Classification information	num_correct	Number of correctly classified instances
	num_incorrect	Number of incorrectly classified instances
	percent_correct	
	percent_incorrect	
Classification information (each value available once for each class value)	area_under_roc	Area under ROC Curve
	f_measure	
	precision	
	recall	
	num_false_positives	
	num_false_negatives	
	num_true_positives	
	num_true_negatives	
	true_negative_rate	
	true_positive_rate	
	false_negative_rate	
	false_positive_rate	
Classification confusion matrix (each value available once for each pair of class values)	confusion_matrix_predicted	Predicted class value
	confusion_matrix_actual	Actual class value
	confusion_matrix_value	Number of instances with above actual/predicted class value
Regression information	root_mean_squared_error	
	mean_absolute_error	
	r_square	
Crossvalidation information	crossvalidation_uri	URI of crossvalidation (if available)
	crossvalidation_fold	Fold of crossvalidation (if available)

4.2 Validation Routines

4.2.1 Single validation with supplied test set

General Information about the service
Service description A prediction model is applied to a test dataset, various performance measures are calculated to determine the quality of the model.
URI http://opentox.informatik.uni-freiburg.de/validation/
Date completed 2009 – December 28
OpenTox API Reference

www.opentox.org/dev/apis/api-1.1/Validation
Partner responsible for the implementation Albert-Ludwigs Universität, Freiburg
Contact within OT Martin Gütlein guetlein@informatik.uni-freiburg.de Michael Rautenberg rautenb@informatik.uni-freiburg.de
Comments (including reviews)

Request/Response Information
Posted Parameters model URI test_dataset URI
Response Validation URI or Task URI Once the model is successfully evaluated, the validation URI (or in case of ongoing calculations, the task URI) is returned to the client within the response and the status is set to 200.

Status Codes
200 Success - The request has succeeded and the new validation URI is returned.
400 Bad Request - Some parameter provided was wrong or some mandatory parameter was not posted (an illegal model or dataset was specified).
404 Not Found - the particular validation was not found
500 Internal Server Error - Validation/prediction error

Implementation Information
HTTP Method POST

Programming Language
Ruby
Libraries used
–

Examples
Example 1
curl -X POST -d model_uri="http://opentox.informatik.uni-freiburg.de/model/1" -d test_dataset_uri="http://opentox.informatik.uni-freiburg.de/dataset/3" -d prediction_feature= "http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome_CPDBAS_Hamster" http://opentox.informatik.uni-freiburg.de/validation

4.2.2 Single validation with model construction from training dataset and validation on a supplied test set

General Information about the service
Service description
This service allows to construct a model based on a supplied training dataset and algorithm (including the algorithm parameters). The model is then evaluated on the supplied test dataset.
URI
http://opentox.informatik.uni-freiburg.de/validation/
Date completed
2009 – December 28
OpenTox API Reference
http://www.opentox.org/dev/apis/api-1.1/Validation
Partner responsible for the implementation
Albert-Ludwigs Universität, Freiburg
Contact within OT
Martin Gütlein guetlein@informatik.uni-freiburg.de Michael Rautenberg rautenb@informatik.uni-freiburg.de

Comments (including reviews)

Request/Response Information

Posted Parameters

algorithm_uri

prediction_feature

algorithm_params (string, default="")

training_dataset_uri

test_dataset_uri

y_scramble (boolean, default=false)

y_scramble_seed (integer, default=1)

Response

Validation URI or Task URI

Once the model is successfully constructed and evaluated, the validation URI (or in case of ongoing calculations, the task URI) is returned to the client within the response and the status is set to 200.

Status Codes

200

Success – The request has succeeded and the new validation URI is returned.

400

Bad Request – Some parameter provided was wrong or some mandatory parameter was not posted (an illegal algorithm or dataset was specified).

404

Not Found – the particular validation was not found

500

Internal Server Error – Validation/prediction error

Implementation Information

HTTP Method

POST

Programming Language

Ruby
Libraries used
–

Examples
Example 1
curl -X POST -d algorithm_uri="http://opentox.informatik.uni-freiburg.de/algorithm/lazar" -d training_dataset_uri="http://opentox.informatik.uni-freiburg.de/dataset/2" -d test_dataset_uri="http://opentox.informatik.uni-freiburg.de/dataset/3" -d prediction_feature= "http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome_CPDBAS_Hamster" -d algorithm_params="feature_generation_uri=http://opentox.informatik.uni-freiburg.de/algorithm/fminer" http://opentox.informatik.uni-freiburg.de/validation

4.2.3 Single validation with model construction using a training and testing split from a supplied dataset

General Information about the service
Service description
This service allows constructing a model based on a split of the supplied dataset and algorithm (including the algorithm parameters). The model is then evaluated on the remainder of the dataset.
URI
http://opentox.informatik.uni-freiburg.de/validation/training_test_split
Date completed
2009 – December 28
OpenTox API Reference
http://www.opentox.org/dev/apis/api-1.1/Validation
Partner responsible for the implementation
Albert-Ludwigs Universität, Freiburg
Contact within OT
Martin Gütlein guetlein@informatik.uni-freiburg.de Michael Rautenberg rautenb@informatik.uni-freiburg.de

Comments (including reviews)

Request/Response Information

Posted Parameters

algorithm_uri

prediction_feature

algorithm_params (string, default="")

dataset_uri

split_ratio(float, default=0.66)

random_seed(integer, default=1)

y_scramble (boolean, default=false)

y_scramble_seed (integer, default=1)

Response

Validation URI or Task URI

Once the model is successfully evaluated, the validation URI (or in case of ongoing calculations, the task URI) is returned to the client within the response and the status is set to 200.

Status Codes

200

Success – The request has succeeded and the new validation URI is returned.

400

Bad Request – Some parameter provided was wrong or some mandatory parameter was not posted (an illegal algorithm or dataset was specified).

404

Not Found – the particular validation was not found

500

Internal Server Error – Validation/prediction error

Implementation Information

HTTP Method

POST

Programming Language	
Ruby	
Libraries used	
–	

Examples

Example 1

```
curl -X POST -d algorithm_uri="http://opentox.informatik.uni-freiburg.de/algorithm/lazar" -
d dataset_uri="http://opentox.informatik.uni-freiburg.de/dataset/1" -d prediction_feature
="http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome_CPDBAS_Hamster"
-d algorithm_params="feature_generation_uri=http://opentox.informatik.uni-
freiburg.de/algorithm/fminer" -d split_ratio=0.9 -d random_seed=2
http://opentox.informatik.uni-freiburg.de/validation/training_test_split
```

4.3 Cross-Validation Result

When performing a *k*-fold cross-validation, the validation service internally performs *k* training-test-set validations on the respective folds. Therefore, a cross-validation object contains links to k validations. All values of a cross-validation object are listed in the table below (see OpenTox Ontology[15] for more details).

Value	Description
algorithm_uri	URI of the validated algorithm
num_folds	Number of folds
random_seed	Random seed, determines random split into folds (to make it reproducible and to guarantee a fair comparison between cross-validations)
stratified	Boolean value, true means that the class values have been equally distributed through the folds
dataset_uri	URI of dataset used for validation
validation_uris	List with k validation URIs, one for each fold

[15] www.opentox.org/data/documents/development/RDF files/OpenToxOntology/view

4.4 Cross-Validation Routines

4.4.1 *K*-fold cross-validation with supplied dataset set

General Information about the service
Service description
Performs a *k*-fold cross-validation, i.e. the supplied dataset is split into *k* subsets and *k-1* subsets are used to generate a model by the specified algorithm and the remaining subset is used for evaluation on the models performance.
URI
http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/
Date completed
2009 – December 28
OpenTox API Reference
http://www.opentox.org/dev/apis/api-1.1/Validation
Partner responsible for the implementation
Albert-Ludwigs Universität, Freiburg
Contact within OT
Martin Gütlein guetlein@informatik.uni-freiburg.de Michael Rautenberg rautenb@informatik.uni-freiburg.de
Comments (including reviews)

Request/Response Information
Posted Parameters
algorithm_uri prediction_feature algorithm_params (string, default="") num_folds (integer, default=10) random_seed (integer, default=1) stratified (boolean, default=true) y_scramble (boolean, default=false) y_scramble_seed (integer, default=1)

Response

Cross-validation URI or Task URI

Once the model is successfully evaluated, the validation URI (or in case of ongoing calculations, the task URI) is returned to the client within the response and the status is set to 200.

Status Codes

200

Success - The request has succeeded and the new cross-validation URI is returned.

400

Bad Request - Some parameters provided were wrong or some mandatory parameters were not posted (an illegal algorithm and/or dataset was specified).

404

Not Found - the particular cross-validation was not found

500

Internal Server Error - Cross-validation/prediction error

Implementation Information

HTTP Method

POST

Programming Language

Ruby

Libraries used

-

Examples

Example 1

```
curl -X POST -d algorithm_uri="http://opentox.informatik.uni-freiburg.de/algorithm/lazar"
 -d dataset_uri="http://opentox.informatik.uni-freiburg.de/dataset/1"
 -d prediction_feature =
"http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome_CPDBAS_Hamster"
-d algorithm_params="feature_generation_uri=http://opentox.informatik.uni-
freiburg.de/algorithm/fminer"  -d num_folds=5  -d random_seed=2  -d stratified=false
```

http://opentox.informatik.uni-freiburg.de/validation/crossvalidation

4.4.2 Leave-one-out cross-validation

General Information about the service
Service description This service is similar to 4.2.1. The main difference is that for a leave-one-out cross-validation k=|D|, where D is the supplied dataset. So, every entry in the dataset is predicted once using a model constructed by the rest of the dataset employing a specified algorithm.
URI http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/loo
Date completed 2009 – December 28
OpenTox API Reference http://www.opentox.org/dev/apis/api-1.1/Validation
Partner responsible for the implementation Albert-Ludwigs Universität, Freiburg
Contact within OT Martin Gütlein guetlein@informatik.uni-freiburg.de Michael Rautenberg rautenb@informatik.uni-freiburg.de
Comments (including reviews)

Request/Response Information
Posted Parameters algorithm_uri prediction_feature algorithm_params (string, default="") y_scramble (boolean, default=false) y_scramble_seed (integer, default=1)
Response Validation URI or Task URI

Once the model is successfully evaluated, the validation URI (or in case of ongoing calculations, the task URI) is returned to the client within the response and the status is set to 200.

Status Codes	
200	Success – The request has succeeded and the new validation URI is returned.
400	Bad Request – Some parameter provided was wrong or some mandatory parameter was not posted (an illegal model or dataset was specified).
404	Not Found – the particular validation was not found
500	Internal Server Error – Validation/prediction error

Implementation Information	
HTTP Method	POST
Programming Language	Ruby
Libraries used	–
Examples	
Example 1	

4.5 Reporting Routines

4.5.1 Validation Report

General Information about the service
Service description
A report on a validation creates a DocBook XML document in the background, which can be converted into a number of different formats such as HTML, PDF, RTF and others. The validation report includes information such as an overview of the results, ROC plots for each individual class, the confusion matrix, all calculated results, as well as the actual predictions for each of the supplied entries in the dataset. To generate a report, only the validation resource has to be submitted.
URI
http://opentox.informatik.uni-freiburg.de/validation/report/validation
Date completed
2009 – December 28
OpenTox API Reference
http://www.opentox.org/dev/apis/api-1.1/Validation
Partner responsible for the implementation
Albert-Ludwigs Universität, Freiburg
Contact within OT
Martin Gütlein guetlein@informatik.uni-freiburg.de Michael Rautenberg rautenb@informatik.uni-freiburg.de
Comments (including reviews)

Request/Response Information
Posted Parameters
Validation URI
Response
Report URI or Task URI

Status Codes

200	
	Success - The request has succeeded and the new report URI is returned.
400	
	Bad Request - Some parameter provided was wrong or some mandatory parameter was not posted.
404	
	Not Found - the particular report or report type was not found
500	
	Internal Server Error - an error occurred during creation of the report

Implementation Information	
HTTP Method	
	POST
Programming Language	
	Ruby
Libraries used	
	R (RinRuby)

Examples	
Example 1	
	curl -X POST -d validation_uris="http://opentox.informatik.uni-freiburg.de/validation/1" http://opentox.informatik.uni-freiburg.de/validation/report/validation

4.5.2 Cross-Validation Report

General Information about the service	
Service description	
	A report on a cross-validation creates a DocBook XML document in the background, which can be converted into a number of different formats such as HTML, PDF, RTF and others. The internal structure is that it is comprised of a number of single validation reports and includes a summary of those, as well as all individual results. To generate a report, only the cross-validation resource has to be submitted.

URI http://opentox.informatik.uni-freiburg.de/validation/report/crossvalidation
Date completed 2009 – December 28
OpenTox API Reference http://www.opentox.org/dev/apis/api-1.1/Validation
Partner responsible for the implementation Albert-Ludwigs Universität, Freiburg
Contact within OT Martin Gütlein guetlein@informatik.uni-freiburg.de Michael Rautenberg rautenb@informatik.uni-freiburg.de
Comments (including reviews)

Request/Response Information
Posted Parameters Validation URI
Response Report URI or Task URI

Status Codes
200 Success – The request has succeeded and the new report URI is returned.
400 Bad Request – Some parameter provided was wrong or some mandatory parameter was not posted.
404 Not Found – the particular report or report type was not found
500 Internal Server Error – an error occurred during creation of the report

Implementation Information

HTTP Method

POST

Programming Language

Ruby

Libraries used

R (RinRuby)

Examples

Example 1

```
curl -X POST -d validation_uris="http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/1"  http://opentox.informatik.uni-freiburg.de/validation/report/crossvalidation
```

4.6 Example Reports

In the appendix 6.1 two example reports are given.

4.6.1 Validation report

The first example in Appendix 6.1.1 is a validation report from a train and test validation. The input to the system was a training set, a test set, and the algorithm including the algorithm parameters. With this input, a model using the Lazar algorithm was constructed for the hamster carcinogenicity endpoint. The report contains the initial parameters as submitted by the user, global performance parameters calculated using the model, a ROC plot, and the actual predictions of the model.

4.6.2 Cross-Validation report

The example in Appendix 6.1.2 is a cross-validation report using five folds, again for predicting the hamster carcinogenicity endpoint. Again a number of global measures are calculated and displayed. The ROC plot of each of the five generated models is aggregated into one single plot as well as the average ROC plot displayed. The overall confusion matrix is given together with all predictions.

4.7 Future Work

4.7.1 Interoperability with partner web services

The validation routines listed above have been successfully applied to web services provided by In Silico Toxicology (IST). The Lazar algorithm[16] was used to make predictions, based on features calculated with the Fminer[17] web service. A current issue is to integrate further web services provided by other OpenTox partners.

4.7.2 Graphical user interface for validation

The described validation routines can be tested by directly executing the REST commands (e.g. with the command line tool curl, see 3.5). A graphical, more user-friendly access to

[16] github.com/helma/opentox-algorithm/

[17] github.com/helma/opentox-algorithm/

this functionality will soon be provided by the ToxCreate[18] application (See OpenTox Deliverable D2.2 for details).

4.7.3 Technical issues

With the current implementation being an initial prototype, there is still work to do in order to provide a more reliable web application. Continuous automated testing has revealed deficits when processing numerous validation requests simultaneously, whose performance improvement is the subject of current work.

[18] www.toxcreate.org

5. Conclusions

This report summarizes the work that has been accomplished within the OpenTox Framework regarding the development of the initial prototype of validation and reporting routines for predictive toxicology modelling. One of the main decisions in the initial OpenTox Framework application development was to implement all web services based on the REST architecture, allowing for independent development and deployment of components combined with effective intra-component communication. These advantages allow for future validation and reporting routines to be integrated with minimal effort. Currently we have implemented within OpenTox basic validation routines, simple validation (with supplied test set or training/test split), cross-validation routines (including leave-one-out), as well as making initial reporting routines available. The validation web service was integrated into the OpenTox-based ToxCreate application[19] to validate predictive toxicology models created by ToxCreate users.

[19] www.opentox.org/toxicity-prediction/

6. Appendix

6.1 Validation Report Examples

This section gives two validation reports as example, a simple training test-set validation report, as well as a cross-validation report.

Please take notice that the design of the reports is still very basic. The arrangement of the content, table design, plots, and overall formatting is work-in-progress.

6.1.1 Validation report

Created at 02.19.2010 - 16:38

Table of Contents

Results

This section contains results.

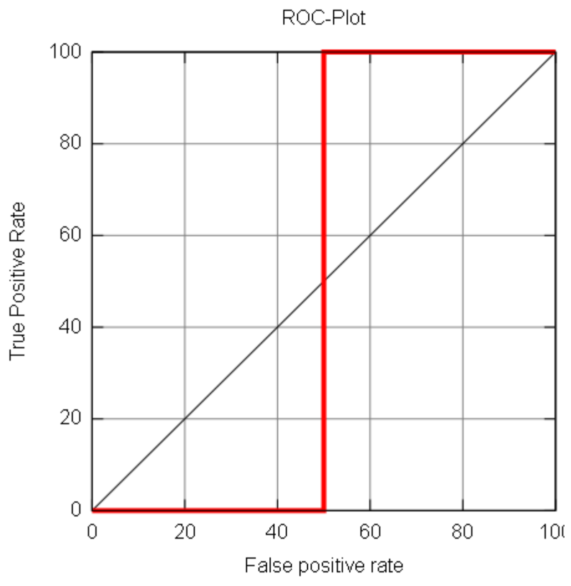
Table 1. Results

model_uri	http://www.opentox.org/temp/model/1
training_dataset_uri	http://www.opentox.org/temp/dataset/3
test_dataset_uri	http://www.opentox.org/temp/dataset/2
prediction_feature	http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcom
area_under_roc	true: 0.50, false: 1.00
percent_correct	50.00
true_positive_rate	true: 1.00, false: 0.50
true_negative_rate	true: 0.50, false: 1.00

Roc Plot

This section contains the roc plot.

Figure 1. Roc Plot for all classes



Confusion Matrix

This section contains the confusion matrix.

Table 2. Confusion Matrix

		actual		
		true	false	total
predicted	true	2	2	4
	false	0	2	2
	total	2	4	

All Results

This section contains results.

Table 3. All Results

id	1
uri	http://www.opentox.org/temp/example.org/1
model_uri	http://www.opentox.org/temp/model/1
training_dataset_uri	http://www.opentox.org/temp/dataset/3
prediction_feature	http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcom

test_dataset_uri	http://www.opentox.org/temp/dataset/2
prediction_dataset_uri	http://www.opentox.org/temp/dataset/5
finished	true
created_at	2010-02-19T16:37:07+01:00
real_runtime	1.82
num_instances	8
num_without_class	0
percent_without_class	0
num_unpredicted	2
percent_unpredicted	25
num_correct	4
num_incorrect	2
percent_correct	50.00
percent_incorrect	25.00
area_under_roc	true: 0.50, false: 1.00
false_negative_rate	true: 0.00, false: 0.50
false_positive_rate	true: 0.50, false: 0.00
f_measure	true: 0.67, false: 0.67
num_false_positives	true: 2, false: 0
num_false_negatives	true: 0, false: 2
num_true_positives	true: 2, false: 2
num_true_negatives	true: 2, false: 2
precision	true: 0.50, false: 1.00
recall	true: 1.00, false: 0.50
true_negative_rate	true: 0.50, false: 1.00
true_positive_rate	true: 1.00, false: 0.50
confusion_matrix	confusion_matrix_predicted: true, confusion_matrix_actual: true: 2

Predictions

This section contains predictions.

Table 4. Predictions

compound	actual value	predicted value	missclassified	confidence value
http://localhost:4000/InChI=1S/C12H9NO2/c14-13(15)11-7-6-9-5-4-8-	false	false		0.07
http://localhost:4000/InChI=1S/C5H12N2O4/c8-2-1-7(6-11)3-5(10)4-9	false	true	X	0.14
http://localhost:4000/InChI=1S/C14H19N3S.ClH/c1-16(2)9-10-17(12-1	false	true	X	0.02
http://localhost:4000/InChI=1S/C9H7N3O4S/c1-5(13)10-9-11-6(4-17-9	true	true		0.08
http://localhost:4000/InChI=1S/C12H8Cl6O/c13-8-9(14)11(16)5-3-1-2	false			
http://localhost:4000/InChI=1S/C8H5N3O4S/c12-4-9-8-10-5(3-16-8)6-	true	true		0.08
http://localhost:4000/InChI=1S/C3H6ClNO/c1-5(2)3(4)6/h1-2H3	true			
http://localhost:4000/InChI=1S/C27H30O16/c1-8-17(32)20(35)22(37)2	false	false		0.09

6.1.2 Cross-validation Report

Created at 02.19.2010 - 16:38

Table of Contents

Mean Results

This section contains results.

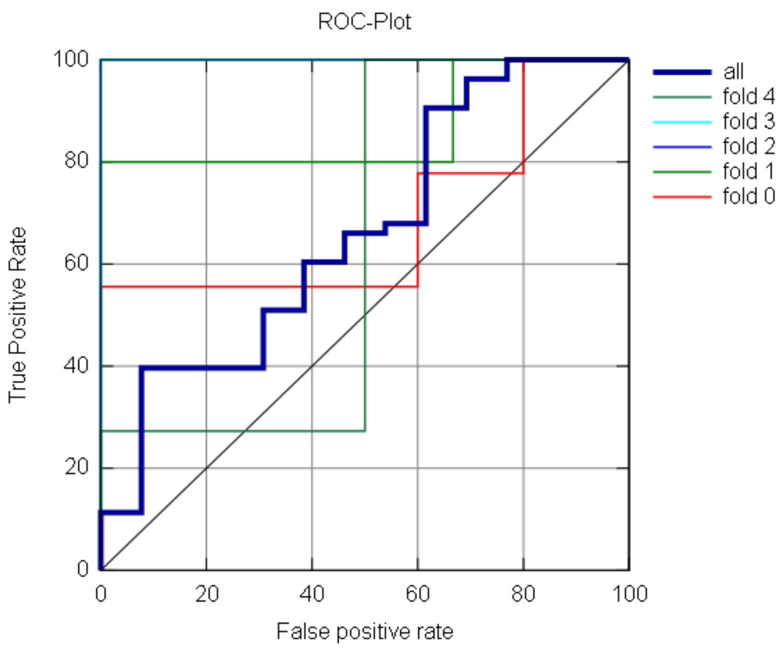
Table 1. Mean Results

algorithm_uri	http://www.opentox.org/temp/algorithm/lazar
dataset_uri	http://www.opentox.org/temp/dataset/1
num_folds	5.00
area_under_roc	true: 0.88, false: 0.67 +- true: 0.03, false: 0.17
percent_correct	62.35 +- 44.98
true_positive_rate	true: 0.84, false: 0.77
true_negative_rate	true: 0.77, false: 0.84

Roc Plot

This section contains the roc plot.

Figure 1. Roc Plot for all classes



Confusion Matrix

This section contains the confusion matrix.

Table 2. Confusion Matrix

		actual		
		true	false	total
predicted	true	32	7	39
	false	7	21	29
	total	39	29	

Results

algorithm_uri	http://www.opentox.org/temp/algorithm/lazar	http://www.opentox.org/temp/algorithm/lazar	http://www.opentox.org/temp/algorithm/lazar	http://www.opentox.org/temp/algorithm/lazar	http://www.opentox.org/temp/algorithm/lazar
dataset_uri	http://www.opentox.org/temp/dataset/1	http://www.opentox.org/temp/dataset/1	http://www.opentox.org/temp/dataset/1	http://www.opentox.org/temp/dataset/1	http://www.opentox.org/temp/dataset/1
crossvalidation_fold	1	2	3	4	5
area_under_roc	true: 0.75, false: 0.60	true: 1.00, false: 0.00	true: 1.00, false: 1.00	true: 1.00, false: 0.75	true: 0.67, false: 1.00

percent_correct	52.94	58.82	70.59	64.71	64.71
true_positive_rate	true: 0.57, false: 0.71	true: 0.86, false: 0.67	true: 0.90, false: 0.75	true: 0.88, false: 1.00	true: 1.00, false: 0.71
true_negative_rate	true: 0.71, false: 0.57	true: 0.67, false: 0.86	true: 0.75, false: 0.90	true: 1.00, false: 0.88	true: 0.71, false: 1.00

This section contains results.

Table 3. Results

All Results

This section contains results.

Table 4. All Results

id	2	3	4
uri	http://example.opentox.org/2	http://example.opentox.org/3	http://example.org/4
model_uri	http://www.opentox.org/temp/model/2	http://www.opentox.org/temp/model/3	http://www.opentox.org/temp/model/4
training_dataset_uri	http://www.opentox.org/temp/dataset/6	http://www.opentox.org/temp/dataset/8	http://www.opentox.org/temp/dataset/10
prediction_feature	http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcom	http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcom	http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcom
test_dataset_uri	http://www.opentox.org/temp/dataset/7	http://www.opentox.org/temp/dataset/9	http://www.opentox.org/temp/dataset/11
prediction_dataset_uri	http://www.opentox.org/temp/dataset/17	http://www.opentox.org/temp/dataset/19	http://www.opentox.org/temp/dataset/21
finished	true	true	true
created_at	2010-02-19T16:37:20+01:00	2010-02-19T16:37:22+01:00	2010-02-19T16:37:24+01:00
real_runtime	2.33	1.99	2.11
num_instances	17	17	17
num_without_class	0	0	0
percent_without_class	0	0	0
num_unpredicted	3	4	3
percent_unpredicted	17	23	17
num_correct	9	10	12

num_incorrect	5	3	2
percent_correct	52.94	58.82	70.59
percent_incorrect	29.41	17.65	11.76
area_under_roc	true: 0.75, false: 0.60	true: 1.00, false: 0.00	true: 1.00, false: 1.00
false_negative_rate	true: 0.43, false: 0.29	true: 0.14, false: 0.33	true: 0.10, false: 0.25
false_positive_rate	true: 0.29, false: 0.43	true: 0.33, false: 0.14	true: 0.25, false: 0.10
f_measure	true: 0.62, false: 0.67	true: 0.80, false: 0.73	true: 0.90, false: 0.75
num_false_positives	true: 2, false: 3	true: 2, false: 1	true: 1, false: 1
num_false_negatives	true: 3, false: 2	true: 1, false: 2	true: 1, false: 1
num_true_positives	true: 4, false: 5	true: 6, false: 4	true: 9, false: 3
num_true_negatives	true: 5, false: 4	true: 4, false: 6	true: 3, false: 9
precision	true: 0.67, false: 0.62	true: 0.75, false: 0.80	true: 0.90, false: 0.75
recall	true: 0.57, false: 0.71	true: 0.86, false: 0.67	true: 0.90, false: 0.75
true_negative_rate	true: 0.71, false: 0.57	true: 0.67, false: 0.86	true: 0.75, false: 0.90
true_positive_rate	true: 0.57, false: 0.71	true: 0.86, false: 0.67	true: 0.90, false: 0.75
confusion_matrix	confusion_matrix_predicted: true, confusion_matrix_actual: true: 4	confusion_matrix_predicted: true, confusion_matrix_actual: true: 6	confusion_matrix_predicted: true, confusion_matrix_actual: true: 9

Predictions

This section contains predictions.

Table 5. Predictions

crossvalidation_fold	compound	actual value	predicted value	missclassified	confidence value
1	http://localhost:4000/InChI=1S/C12H12N2O3/c1-2-12(8-6-4-3-5-7-8)9	false	false		0.05
1	http://localhost:4000/InChI=1S/C10H13N3O2/c1-13(12-15)7-3-5-10(14	false	true	X	0.09
1	http://localhost:4000/InChI=1S/C4H6N2O3/c1-3-2-6(5-8)4(7)9-3/h3H,	true	true		0.14
1	http://localhost:4000/InChI=1S/C20H19N3.ClH/c1-13-12-16(6-11-19(1	false	false		0.12
1	http://localhost:4000/InChI=1S/C2HCl3/c3-1-2(4)5/h1H	false			
1	http://localhost:4000/InChI=1S/C15H13NO/c1-10(17)16-13-6-7-	true	false	X	0.11

crossvalidation_fold	compound	actual value	predicted value	missclassified	confidence value
	15-12(				
1	http://localhost:4000/InChI=1S/C6H12N2O4/c1-5(10)2-8(7-12)3-6(11)	true	true		0.15
1	http://localhost:4000/InChI=1S/HNO2.Na/c2-1-3;/h(H,2,3);/q;+1/p-1	false			
1	http://localhost:4000/InChI=1S/C9H11N3O2/c13-10-12-6-2-4-9(12)8-3	false	true	X	0.12
1	http://localhost:4000/InChI=1S/C20H22N8O5/c1-28(9-11-8-23-17-15(2	false	false		0.10
1	http://localhost:4000/InChI=1S/C6H5N2.BF4/c7-8-6-4-2-1-3-5-6;2-1(	false	false		0.17
1	http://localhost:4000/InChI=1S/C2H8N2.2ClH/c1-3-4-2;;/h3-4H,1-2H3	true	true		0.08
1	http://localhost:4000/InChI=1S/C17H17ClO6/c1-8-5-9(19)6-12(23-4)1	false	false		0.09
1	http://localhost:4000/InChI=1S/C6H14N2O4/c1-5(10)2-8(7-12)3-6(11)	true	true		0.16
1	http://localhost:4000/InChI=1S/C15H13NO2/c1-10(17)16(18)13-6-7-15	true	false	X	0.12
1	http://localhost:4000/InChI=1S/C5H6N2OS/c1-3-2-4(8)7-5(9)6-3/h2H,	true	false	X	0.05
1	http://localhost:4000/InChI=1S/BF4.Na/c2-1(3,4)5;/q-1;+1	false			
2	http://localhost:4000/InChI=1S/C16H13N/c1-2-8-15(9-3-1)17-16-11-1	false	false		0.11
2	http://localhost:4000/InChI=1S/Cd.H2O4S/c;1-5(2,3)4/h;(H2,1,2,3,4	false			
2	http://localhost:4000/InChI=1S/C2H5ClO/c1-4-2-3/h2H2,1H3	true			
2	http://localhost:4000/InChI=1S/C14H14ClN3O2S/c1-8-4-3-5-10(9(8)2)	false	false		0.05
2	http://localhost:4000/InChI=1S/C5H10N2O3/c1-5(9)4-7(6-10)2-3-8/h8	true	true		0.14
2	http://localhost:4000/InChI=1S/C2H3Cl/c1-2-3/h2H,1H2	true			
2	http://localhost:4000/InChI=1S/C2H6O/c1-2-3/h3H,2H2,1H3	false			
2	http://localhost:4000/InChI=1S/C4H7N3O3/c1-3(8)2-7(6-10)4(5)9/h2H	true	true		0.14
2	http://localhost:4000/InChI=1S/C2H8N2O/c3-4-1-2-5/h4-5H,1-3H2	false	true	X	0.08
2	http://localhost:4000/InChI=1S/C19H24N2O2/c22-18-13-20(19(23)15-7	false	false		0.07

crossvalidation_fold	compound	actual value	predicted value	missclassified	confidence value
2	http://localhost:4000/InChI=1S/C5H4O2/c6-4-5-2-1-3-7-5/h1-4H	false	true	X	0.04
2	http://localhost:4000/InChI=1S/C10H13NO2/c1-8(2)13-10(12)11-9-6-4	false	false		0.11
2	http://localhost:4000/InChI=1S/C8H6N4O4S/c13-4-9-11-8-10-5(3-17-8	true	true		0.15
2	http://localhost:4000/InChI=1S/C4H10N2O3/c1-6(5-9)2-4(8)3-7/h4,7-	true	true		0.15
2	http://localhost:4000/InChI=1S/C3H6N2O2/c6-4-5-1-2-7-3-5/h1-3H2	true	true		0.14
2	http://localhost:4000/InChI=1S/H4N2/c1-2/h1-2H2	true	true		0.25
2	http://localhost:4000/InChI=1S/C6Cl6/c7-1-2(8)4(10)6(12)5(11)3(1)	true	false	X	0.11
3	http://localhost:4000/InChI=1S/C6H12N4O2/c1-5-3-9(7-11)4-6(2)10(5	true	true		0.15
3	http://localhost:4000/InChI=1S/C5H8O2/c1-4(2)5(6)7-3/h1H2,2-3H3	false			
3	http://localhost:4000/InChI=1S/C6H7N3O/c7-9-6(10)5-1-3-8-4-2-5/h1	false	true	X	0.00
3	http://localhost:4000/InChI=1S/C5H11N3O3/c1-2-8(7-11)5(10)6-3-4-9	true	true		0.15
3	http://localhost:4000/InChI=1S/C2H6N2O/c1-4(3)2-5/h2H,3H2,1H3	true	true		0.07
3	http://localhost:4000/InChI=1S/C4H8N2O/c7-5-6-3-1-2-4-6/h1-4H2	true	true		0.12
3	http://localhost:4000/InChI=1S/2C2H4O2.4H2O.3Pb/c2*1-2(3)4;;;;;;	false			
3	http://localhost:4000/InChI=1S/C9H11N3O2/c10-9(13)12(11-14)7-6-8-	true	true		0.06
3	http://localhost:4000/InChI=1S/C19H17N3.ClH/c20-16-7-1-13(2-8-16)	false	false		0.08
3	http://localhost:4000/InChI=1S/C9H9NS/c11-8-10-7-6-9-4-2-1-3-5-9/	false	false		0.05
3	http://localhost:4000/InChI=1S/C17H17ClO3/c1-17(2,16(19)20)21-11-	false	false		0.06
3	http://localhost:4000/InChI=1S/C2H8N2/c1-4(2)3/h3H2,1-2H3	true	true		0.07
3	http://localhost:4000/InChI=1S/C4H5Cl/c1-3-4(2)5/h3H,1-2H2	false			
3	http://localhost:4000/InChI=1S/C6H10N2O2/c1-3-4-8(7-10)5-6(2)9/h3	true	true		0.14

crossvalidation_fold	compound	actual value	predicted value	missclassified	confidence value
3	http://localhost:4000/InChI=1S/C9H11N3O/c13-11-12-6-2-4-9(12)8-3-	true	true		0.03
3	http://localhost:4000/InChI=1S/C11H8N2O5/c12-11(14)8(9-2-1-5-17-9	true	false	X	0.02
3	http://localhost:4000/InChI=1S/C6H11N3O3/c1-3-9(8-12)6(11)7-4-5(2	true	true		0.14
4	http://localhost:4000/InChI=1S/CH6N2/c1-3-2/h3H,2H2,1H3	true	true		0.09
4	http://localhost:4000/InChI=1S/C20H22O3/c1-20(2,19(21)22)23-16-12	false	false		0.09
4	http://localhost:4000/InChI=1S/C6H10N2O/c1-3-5-8(7-9)6-4-2/h3-4H,	true	true		0.11
4	http://localhost:4000/InChI=1S/C5H10N2O/c8-6-7-4-2-1-3-5-7/h1-5H2	true	true		0.11
4	http://localhost:4000/InChI=1S/C15H10O7.2H2O/c16-7-4-10(19)12-11(	false	false		0.09
4	http://localhost:4000/InChI=1S/CH2O/c1-2/h1H2	false			
4	http://localhost:4000/InChI=1S/C14H9Cl5/c15-11-5-1-9(2-6-11)13(14	false	false		0.08
4	http://localhost:4000/InChI=1S/C3H6N2O/c6-4-5-2-1-3-5/h1-3H2	true	true		0.11
4	http://localhost:4000/InChI=1S/C3H6O2/c4-1-3-2-5-3/h3-4H,1-2H2	true			
4	http://localhost:4000/InChI=1S/C4H8N2O3/c1-3-9-4(7)6(2)5-8/h3H2,1	true	true		0.08
4	http://localhost:4000/InChI=1S/C10H12ClNO2/c1-7(2)14-10(13)12-9-5	false	false		0.08
4	http://localhost:4000/InChI=1S/H4N2.H2O4S/c1-2;1-5(2,3)4/h1-2H2;(	true	true		0.25
4	http://localhost:4000/InChI=1S/BrHO3.K/c2-1(3)4;/h(H,2,3,4);/q;+1	true			
4	http://localhost:4000/InChI=1S/C7H15N3O/c1-6-4-10(8-11)5-7(2)9(6)	true	true		0.11
4	http://localhost:4000/InChI=1S/Cd.2ClH/h;2*1H/q+2;;/p-2	false			
4	http://localhost:4000/InChI=1S/C14H8Cl4/c15-11-5-1-9(2-6-11)13(14	true	false	X	0.08
4	http://localhost:4000/InChI=1S/C2H4O/c1-2-3/h2H,1H3	true			
5	http://localhost:4000/InChI=1S/C12H9NO2/c14-13(15)11-7-6-9-5-4-8-	false	false		0.05
5	http://localhost:4000/InChI=1S/C6H10ClN3O3/c1-5(11)4-10(9-	true	true		0.17

crossvalidation_fold	compound	actual value	predicted value	missclassified	confidence value
	13)6(12				
5	http://localhost:4000/InChI=1S/C5H12N2O4/c8-2-1-7(6-11)3-5(10)4-9	false	true	X	0.17
5	http://localhost:4000/InChI=1S/C3H7NO2/c1-2-6-3(4)5/h2H2,1H3,(H2,	true			
5	http://localhost:4000/InChI=1S/C2H4N4/c3-2-4-1-5-6-2/h1H,(H3,3,4,	false			
5	http://localhost:4000/InChI=1S/C6H5NO2/c8-6(9)5-1-3-7-4-2-5/h1-4H	false	false		0.03
5	http://localhost:4000/InChI=1S/C14H19N3S.ClH/c1-16(2)9-10-17(12-1	false	true	X	0.00
5	http://localhost:4000/InChI=1S/C9H7N3O4S/c1-5(13)10-9-11-6(4-17-9	true	true		0.07
5	http://localhost:4000/InChI=1S/C7H6O4/c8-5-2-1-4(7(10)11)3-6(5)9/	false	false		0.06
5	http://localhost:4000/InChI=1S/C9H6O2/c10-9-6-5-7-3-1-2-4-8(7)11-	false	false		0.04
5	http://localhost:4000/InChI=1S/C5H13N3O/c1-7(2)4-5-8(3)6-9/h4-5H2	true	true		0.15
5	http://localhost:4000/InChI=1S/C12H8Cl6O/c13-8-9(14)11(16)5-3-1-2	false			
5	http://localhost:4000/InChI=1S/C8H5N3O4S/c12-4-9-8-10-5(3-16-8)6-	true	true		0.07
5	http://localhost:4000/InChI=1S/C6H12N2O2/c1-5-3-8(7-9)4-6(2)10-5/	true	true		0.17
5	http://localhost:4000/InChI=1S/C3H6ClNO/c1-5(2)3(4)6/h1-2H3	true			
5	http://localhost:4000/InChI=1S/C4H8N2O2/c7-5-6-1-3-8-4-2-6/h1-4H2	true	true		0.18
5	http://localhost:4000/InChI=1S/C27H30O16/c1-8-17(32)20(35)22(37)2	false	false		0.04