



Deliverable D5.4

Final validation routines with facilities to validate against confidential data

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 D5.4	
WP/Task:	WP5	
Name	Final validation routines with facilities to validate against confidential data	
Document ID:	OpenTox Deliverable Report WP5 D5.4	
Date:	31 August 2011	
Status:	Final Version	
Organisation:		
Contributors	Andreas Karwath (AK) Martin Gütlein (MG) Barry Hardy Roman Affentranger	ALU-FR ALU-FR DC DC

Distribution:	Public
---------------	--------

Purpose of Document:	To disseminate the final results on procedures for the validation of toxicity data within the OpenTox Framework including the ability to validate confidential data
----------------------	---

Document History:	1 - Initial draft prepared on May 12, 2011 AK 2 - Update of initial draft on May 30, 2011 MG 3 - Added examples and HTML forms Jul 21, 2011 MG 4 - Updated draft, Aug 18, 2011 BH 5 - Updated and Edited draft, Oct 27, 2011, RA 6- Updated draft, Oct 29, 2011, BH
-------------------	--

Table of contents

Table of contents..... **3**

Summary..... **5**

1 Introduction **7**

2 Validation and Reporting Framework **7**

 2.1 Open Source programming tools 7

 2.2 RESTful Web Service Architecture 7

 2.3 OpenTox Validation and Reporting Application Programming Interfaces (APIs) 8

3 Validation and reporting routines **13**

 3.1 Single validation routines..... 13

 3.1.1 Running a single validation example using HTML forms 14

 3.2 Cross-validation 15

 3.3 Comparing validation results 15

4 (Q)SAR reporting for REACH..... **15**

 4.1 (Q)SAR reporting web service for REACH..... 16

 4.2 (Q)SAR reporting editors for REACH 17

 4.2.1 QMRF Editor 18

 4.2.2 QPRF Editor (Q-edit) 19

5 Validation Routines for Confidential Data **25**

 5.1 REACH legislation and confidential data 25

 5.2 Authentication and authorization in OpenTox 26

 5.2.1 Topological description of access control 26

 5.2.2 Managing access 27

 5.2.3 Policy Creation and management..... 29

 5.2.4 Evaluation of the current access control system 30

 5.2.5 OECD Principles 31

 5.3 Use Case description 31

 5.3.1 Use Case implementation 31

6 Validation examples using confidential data..... **32**

 6.1 Validation against confidential data with ToxCreate 33

 6.2 Validation against validation data using distributed web services..... 35

 6.2.1 Login: 36

 6.2.2 Start validation:..... 36

7 Validation against confidential data using standalone version **41**

7.1	Standalone Installation of OpenTox Web Services	41
7.2	Standalone Installation of particular services	42
7.2.1	AMBIT	42
7.2.2	Jaqpot	43
7.2.3	ToxCreate	44
8	Conclusions	44
9	Appendix	45
9.1	Training test split report	45
9.2	Cross-validation report	49
9.3	Algorithm comparison report	55

Summary

OpenTox is supporting the deployment of validation routines for algorithms and models, as well as reporting capabilities for the generation and presentation of results of alternative testing methods including results of relevance to REACH¹. To prevent sensitive information from being accessed or copied by unauthorized users, the OpenTox validation routines allow the validation against confidential data based on a rigorous authentication and authorization strategy. The report-generating component generates reports to present the results of predictions and (Quantitative) Structure Activity Relationship ((Q)SAR) model validations to the user in a structured reporting format. OpenTox reporting formats are guided by standards and templates such as the (Q)SAR Model Reporting Format (QMRF) and the (Q)SAR Prediction Reporting Format (QPRF)², and by OECD validation principles, which specify that to facilitate the consideration of acceptance of a (Q)SAR model for regulatory purposes, it needs to be associated with the OECD Guidelines for (Q)SAR Validation³.

This report describes and documents the final achievements within the OpenTox project with respect to routines for validation and reporting. We provide an overview of the final framework for validation and reporting routines and illustrate the rationale behind their implementation. The validation and reporting framework follows the open source philosophy generally adopted in OpenTox, and has been realized as standardized web services adhering to the OpenTox Application Programming Interfaces (APIs).⁴ Both the validation and the reporting web service APIs are described in detail in this report. We introduce the validation and reporting solutions made available as OpenTox web services and provide examples of their use. The currently available, state-of-the-art validation and reporting services encompass single validation routines such as Training-Test Set validation, Training-Test Split validation, Bootstrapping, Test Set validation, and also cross-validation routines. An additional reporting functionality is offered to compare validation results of different prediction algorithms that have been applied to the same dataset(s). Such a report may help determining whether one of a few of the algorithms perform significantly better than the rest.

We describe the specific perspective of (Q)SAR reporting for REACH, beginning with a sketch of a user workflow resulting in QMRF and QPRF reports for a (Q)SAR model and a prediction. We introduce an initial implementation of a specialized (Q)SAR reporting service for REACH that manages the creation of both QMRF and QPRF reports, collecting information from several OpenTox web services to automatically fill in report content. Since the complete reports cannot be generated automatically, we incorporated specific editors for QMRF and QPRF reports, which we describe in Section 4.2.

The validation and reporting services are embedded within the OpenTox Authorization and Authentication (A&A) strategy. Thus, they can be applied for handling confidential data, and are an important prerequisite for acceptance of the service. We introduce the concept of A&A and describe why handling confidential data is important also in the context of REACH. The implementation of A&A in OpenTox is described in detail and is put in relation to the OECD Principles for (Q)SAR validation. We describe the use case of validating confidential data.

We also provide examples for validating confidential data either within online services or through a local, stand-alone installation of OpenTox services. The validation web service is seamlessly integrated in the web

¹ http://guidance.echa.europa.eu/docs/guidance_document/information_requirements_r6_en.pdf?vers=20_08_08

² <http://tcsweb3.jrc.it>

³ http://www.oecd.org/home/0,2987,en_2649_201185_1_1_1_1_1,00.html

⁴ www.opentox.org/dev/apis/api-1.2

application ToxCreate, which offers a user-friendly graphical user interface (GUI) while providing confidentiality regarding submitted data. A second example is provided describing how to make use of the validation service while ensuring confidentiality using command-line tools. For some users, even these measures might not offer tight enough security. For such cases, we describe how stand-alone versions of OpenTox or individual services can be used to perform the tasks that are routinely done over the Internet in OpenTox.

After drawing conclusions on our implementation of the reporting and validation services especially in the context of REACH, we provide specific example validation reports on a Training-Test Split Validation, on a Cross-Validation, and on an algorithm comparison.

1 Introduction

To facilitate the consideration of acceptance of a (Quantitative) Structure Activity Relationship i.e. (Q)SAR model for regulatory purposes the OECD published guidelines for the validation of (Q)SAR models⁵. The guidelines state that a model should be associated with a defined endpoint, an unambiguous algorithm, a defined domain of applicability, appropriate measures of goodness-of-fit, robustness and predictivity, and a mechanistic interpretation, if possible. Following these principles, OpenTox offers reporting capabilities for the generation and presentation of results of alternative testing methods including validation and reporting results of relevance to REACH⁶. OpenTox reporting formats are guided by standards and templates such as (Q)SAR Model Reporting Format (QMRF) and the (Q)SAR Prediction Reporting Format (QPRF)⁷, resulting in the generation of reports presenting the results of predictions and (Q)SAR model validations to the user in a structured reporting format.

This report describes and documents the final progress which has been achieved within the OpenTox project with respect to the validation and the automatic creation of reporting facilities for the validation of (Q)SAR models and algorithms using toxicology data.

2 Validation and Reporting Framework

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 ruby⁸. Other applications used are also open source, and include e.g. Apache⁹.

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 stateless¹⁰;
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.

⁵ <http://www.oecd.org/dataoecd/33/37/37849783.pdf> [and http://www.oecd.org/officialdocuments/displaydocumentpdf/?cote=env/jm/mono%282007%292&doclanguage=en](http://www.oecd.org/officialdocuments/displaydocumentpdf/?cote=env/jm/mono%282007%292&doclanguage=en)

⁶ http://guidance.echa.europa.eu/docs/guidance_document/information_requirements_r6_en.pdf?vers=20_08_08

⁷ <http://tcsweb3.jrc.it>

⁸ <http://www.ruby-lang.org>

⁹ <http://httpd.apache.org/>

¹⁰ http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

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) representation¹¹, in particular the XML-formatted 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 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 YAML¹² (YAML Ain't Markup Language).

2.3 OpenTox Validation and Reporting Application 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 1.2 (www.opentox.org/dev/apis/api-1.2). 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/crossvalidation/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 DELETE¹³:

Description	Method	URI	Parameters	Result	Status codes
Get all validations	GET	/	[subjectid]	List of validation URIs	200,404
Retrieves a validation representation	GET	/{id}	[subjectid]	Validation representation in one of the supported MIME types	200,404

¹¹ www.w3.org/RDF

¹² www.yaml.org

¹³ www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

Validates a model on a test dataset	POST	/test_set_validation	[subjectid] model_uri test_dataset_uri test_target_dataset_uri (default = test_dataset_uri) prediction_feature (default = dependent variable of model)	Validation URI or Task URI	200,400,404,500
Builds a model on a training dataset and validates it on a test dataset	POST	/training_test_validation	algorithm_uri prediction_feature algorithm_params (string, default="") training_dataset_uri test_dataset_uri test_target_dataset_uri (default = test_dataset_uri) y_scramble (boolean, default=false) y_scramble_seed (integer, default=1) [subjectid]	Validation URI or Task URI	200,400,404,500
Splits a dataset into training and test dataset according to a certain ratio, and performs a validation	POST	/training_test_split	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) [subjectid]	Validation URI or Task URI	200,400,404,500
Performs a bootstrap validation	POST	/bootstrapping	algorithm_uri prediction_feature dataset_params (string, default="") dataset_uri bootstrap_percentage (float, default=0.66) random_seed (integer, default=1) y_scramble (boolean, default=false)	Validation URI or Task URI	200,400,404,500

			y_scramble_seed (integer, default=1) [subjectid]		
Directly perform a validation by specifying test- and prediction dataset	POST	/validate_datasets	prediction_feature test_dataset_uri test_target_dataset_uri (default = test_dataset_uri) prediction_dataset_uri predicted_feature (.i.e feature in prediction dataset) [subjectid]	Validation URI or Task URI	200,400,404, 500
Deletes a validation.	DELETE	/{id}	[subjectid]	-	200,404

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.

Description	Method	URI	Parameters	Result	Status codes
Get all cross-validations	GET	/crossvalidation	[subjectid]	List of crossvalidation URIs	200,404
Retrieves a cross-validation representation	GET	/crossvalidation/{id}	[subjectid]	Cross-Validation in one of the supported MIME types	200,404
Returns all (k) validations that belong to a crossvalidation	GET	/crossvalidation/{id}/validations	[subjectid]	List of validation URIs	200,404
Performs a k-fold cross-validation.	POST	/crossvalidation	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,	Cross-Validation URI or Task URI	200,400,404, 500

			default=false) y_scramble_seed (integer, default=1) [subjectid]		
Performs a leave-one-out cross-validation.	POST	/crossvalidation/lo o	algorithm_uri prediction_feature algorithm_params (string, default="") y_scramble (boolean, default=false) y_scramble_seed (integer, default=1) [subjectid]	Cross-Validation URI or Task URI	200,400,404, 500
Deletes a cross-validation.	DELETE	/crossvalidation/{id }	[subjectid]	-	200,404

A similar architectural concept was applied to the construction of the API for the **(Q)SAR REACH reporting web service API**, which provides reporting capabilities for all validation objects.

Description	Method	URI	Parameters	Result	Status codes
Create QMRF report	POST	/reach_report/qmrf	application/x-form-www-urlencoded model_uri =Model URI or application/qmrf-xml for creating a report with predefined QMRF XML content	Report URI or Task URI	200,400,404, 500
Replaces QMRF report	POST	/reach_report/qmrf /{reportid}	application/qmrf-xml for creating a report with predefined QMRF XML content	Report URI	200,400,404, 500
Update partially QMRF report	PUT	/reach_report/qmrf /{reportid}	validation_uri = a List of crossvalidation URIs and/or validation URIs of the same model <report_section (as defined in qmrf.dtd)> = content as string	Report URI or Task URI	200,400,404, 500
Delete QMRF report	DELETE	/reach_report/qmrf /{reportid}	deletes the report		

Retrieves the report	GET	/reach_report/qmrf/{reportid}	retrieves the report	representation, , format specified by MIME type (XML, RDF, HTML, PDF, XLS, where applicable)	
Start qmrf editor with report	GET	/reach_report/qmrf/{reportid}/editor	-	return jnlp, starts QMRF editor as Java webstart application	200,404
Create QPRF report	POST	/reach_report/qprf	application/x-form-www-urlencoded model_uri = Model URI One of { dataset_uri = Dataset URI compound_uri = compound uri }, specifying the compounds or application/qprf-format-to-be-defined for creating a report with predefined QPRF content	Report URI or Task URI	200,400,404, 500
Replaces QPRF report	POST	/reach_report/qprf/{reportid}	same as above, replaces the content	Report URI	
Updates QPRF report	PUT	/reach_report/qprf/{reportid}	same as above, but adds new content to the report	Report URI	
Deletes QPRF report	DELETE	/reach_report/qprf/{reportid}	deletes the report		
Retrieves the report content	GET	/reach_report/qprf/{reportid}	retrieves the report	representation, , format specified by MIME type (XML, RDF, HTML, PDF, XLS, where applicable)	
Report searching facilities	GET	/reach_report/{type}	application/x-form-www-urlencoded any or subset of	Retrieves list of reports, related to the model,	

model_uri = Model URI	specified by
dataset_uri = Dataset URI	any of the
compound_uri = Compound URI	parameter URI
algorithm_uri = Algorithm URI	
endpoint_uri = endpoint URI, as defined by the ontology	
search = any free text, etc.	

More information about the validation and reporting API is available at the address <http://opentox.org/dev/apis/api-1.2/Validation>.

3 Validation and reporting routines

The validation service evaluates the performance of prediction algorithms. This is done by building models with training datasets, and applying those prediction models to test datasets. The predicted values are compared to the actual known outcome. As described in the following sections, several state-of-the-art validation and reporting routines are available for both regression and classification models.¹⁴

3.1 Single validation routines

Single validation routines judge the predictive performance achieved on a single test dataset. Several techniques are supported:

- Training test set validation¹⁵
The user has to specify training and test dataset. A predictive model is build based on the training dataset, and applied to the test dataset. This is a common use case, as publicly available datasets are often already separated into training and test datasets.
- Training test split validation¹⁶
The user has to specify a dataset. This dataset is split automatically into training and test dataset (The split is performed randomly; however, the random seed can be set by the user to repeat the exact same split.) A training test set validation is then performed with the two datasets.
- Bootstrapping¹⁷
This is a state-of-the art mechanism that splits a single dataset into training and test dataset via sampling¹⁸. Again a training test set validation is performed with the created datasets.
- Test set validation¹⁹
A test set validation is started with an already existing prediction model, applied to a test dataset.

¹⁴ This section provides an overview of existing validation methods. More details are provided in the API section as well as in previous Deliverable documents 5.1, 5.2 and 5.3.

¹⁵ http://opentox.informatik.uni-freiburg.de/validation/training_test_validation

¹⁶ http://opentox.informatik.uni-freiburg.de/validation/training_test_split

¹⁷ <http://opentox.informatik.uni-freiburg.de/validation/bootstrapping>

¹⁸ http://en.wikipedia.org/wiki/Bootstrapping_%28machine_learning%29

¹⁹ http://opentox.informatik.uni-freiburg.de/validation/test_set_validation

3.1.1 Running a single validation example using HTML forms

All validation routines can be started by sending a REST POST call to the particular service. To make this process easier for users, HTML forms are provided for all validation methods. We exemplify this by performing a training test split validation²⁰. Figure 1 shows the HTML form provided for a training test split validation. The forms are not aimed for a novice user, as the user still has to fill in the URIs pointing to algorithms and datasets. An easier way to use the validation services is shown in Section 6.1. The settings in this example will use Lazar to build a model on 66% of the referenced Salmonella Mutagenicity dataset, and use this model to predict the remaining 34% percent of compounds.

POST command

<code>algorithm_uri:*</code>	<input type="text" value="http://opentox.informatik.uni-freiburg.de/algorithm/lazar"/>
<code>dataset_uri:*</code>	<input type="text" value="http://opentox.informatik.uni-freiburg.de/dataset/333"/>
<code>prediction_feature:*</code>	<input type="text" value="opentox.informatik.uni-freiburg.de/dataset/333/feature/SAL"/>
<code>algorithm_params:</code>	<input type="text"/>
<code>random_seed:</code>	<input type="text" value="1"/>
<code>split_ratio:</code>	<input type="text" value="0.66"/>

Figure 1: Use HTML form to invoke a training test split validation

As soon as the task is finished, a validation URI²¹ will be returned to the user. When visiting the validation URI with a browser, the user can follow a second link in order to ‘Search for [a] corresponding report’²². This page provides an already created report for this validation, as well as another HTML form (Figure 2) to build the validation report.

POST command

<code>validation_uri:*</code>	<input type="text" value="http://opentox.informatik.uni-freiburg.de/validation/451"/>
-------------------------------	---

Figure 2: Use HTML form to build a report for the validation

By pressing ‘Create validation report’ the link to report is returned. A copy of this report can be found in the appendix (section 9.1).

²⁰ Visit http://opentox.informatik.uni-freiburg.de/validation/training_test_split with a browser.

²¹ <http://opentox.informatik.uni-freiburg.de/validation/451>

²² <http://opentox.informatik.uni-freiburg.de/validation/report/validation?validation=http://opentox.informatik.uni-freiburg.de/validation/451>

3.2 Cross-validation

A k-fold cross-validation evaluates a prediction algorithm by splitting a dataset into k folds. The following procedure is then repeated k-times: one fold of the data is used as test-dataset while the remainder is used as training dataset. This method has the advantage of employing the whole dataset as test dataset, and therefore gives a more reliable estimate than a single validation. The validation service provides a k-fold cross-validation routine²³ as well as leave-one-out cross-validation²⁴. The latter is a special case of cross-validation where k is set to the number of compounds in the datasets. It is especially suited for very small datasets.

Similar to the example that was described in the section above (3.1.1), HTML forms can be used to run a cross-validation. Hence, we used Lazar to perform a cross-validation on the Fish toxicity regression dataset²⁵. The corresponding report is attached to the appendix (9.2).

3.3 Comparing validation results

One additional report is provided to compare the validation results of various prediction algorithms that have been applied to the same dataset (or a range of datasets). This report compares important validation statistics of interest, and automatically performs statistical tests to determine if some algorithms are significantly better than others.

We provide an example that compares two state-of-the-art machine learning algorithms: a decision tree algorithm versus a support vector machine. Both prediction algorithms have been evaluated via 5-fold cross-validation on a Blood-Brain-Barrier dataset. The resulting report can be found in the appendix (see 9.3).

4 (Q)SAR reporting for REACH

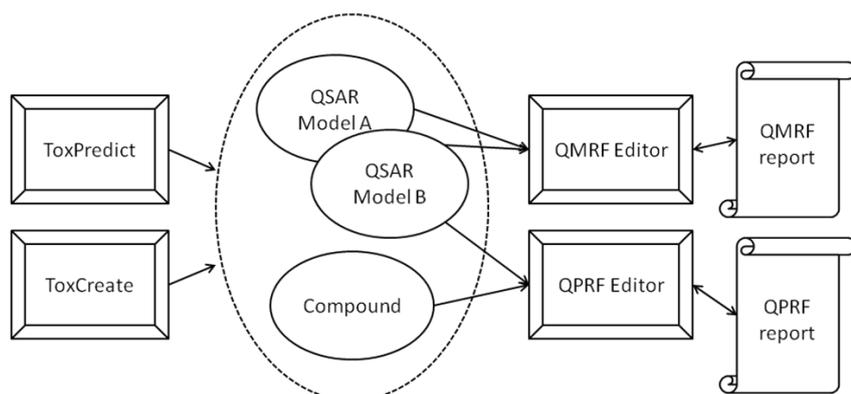


Figure 3: User perspective on (Q)SAR reports for REACH

Figure 3 shows the user perspective on working with QMRF and QPRF reports within the OpenTox framework. The workflow for reports allows OpenTox applications such as ToxCreate²⁶ and ToxPredict²⁷ to easily create

²³ <http://opentox.informatik.uni-freiburg.de/validation/crossvalidation>

²⁴ <http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/loo>

²⁵ Uploaded to our services with dataset-URI: <http://opentox.informatik.uni-freiburg.de/dataset/556>

²⁶ www.toxcreate.net

²⁷ www.toxpredict.net

and access reports. The reports can be created directly from the respective resources the user is working with within the applications: QMRF reports are created from models, QPRF reports are created from predictions, i.e. from a combination of models and compounds. For example, in ToxCreat a QMRF report is created automatically after a new model is built and validated. The user can edit, save, and export this report with the QMRF editor. Similarly, a QPRF editor is available for QPRF reports²⁸. Both editors are implemented as standalone applications that can be started with a web browser. The actual creation of the report is done with a separate OpenTox web service running in the background.

4.1 (Q)SAR reporting web service for REACH

An OpenTox (Q)SAR reporting web service was designed to manage QMRF and QPRF reports for REACH submission purposes. The initial implementation of the service is available at opentox.informatik.uni-freiburg.de/validation/reach_report.

Figure 4 shows how the creation of a report works. A QMRF report is created from an existing (Q)SAR model that is provided to the web service as a URI parameter. The web service internally collects information from a range of other web services to automatically fill in the report content. For example, it queries the validation web service to add all cross validations that have been performed for the algorithm and training dataset (that have been used for building the model). The created QMRF report is stored at the report service. When creating a QPRF report, the compounds which are predicted by the model are required as additional input parameters²⁹.

Like all OpenTox resources, each report is identified and can be accessed via its URI. The report is made available in the official xml format³⁰, as well as in RDF xml (which is the common data exchange format within the OpenTox framework).

The web service furthermore allows the user to update and delete existing reports. The following sections describe the web service functionality in more details: the Application Programming Interface (API) definition for the service is presented in section 3.3.

²⁸ Under the name of Q-edit.

²⁹ The QPRF service implementation is an ongoing development

³⁰ The Document Type Definition (DTD) for the QMRF xml can be found at ambit.sourceforge.net/gmrf/gmrf.dtd; an official xml format for QPRF has yet to be defined.

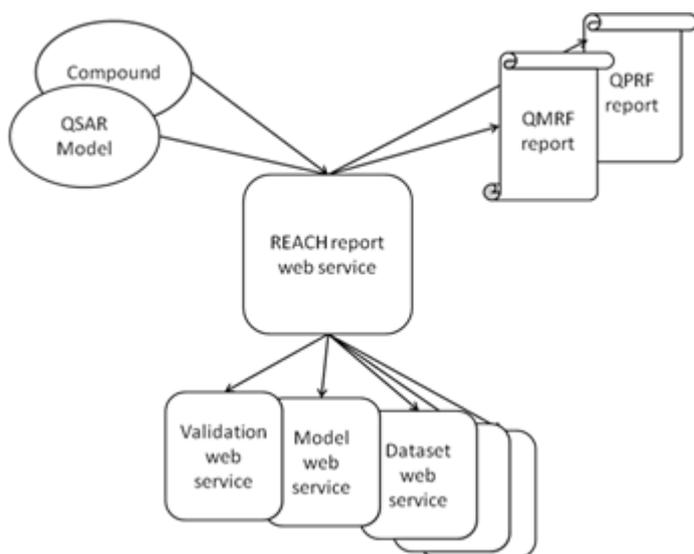


Figure 4: A web service for creating reports

4.2 (Q)SAR reporting editors for REACH

The complete content of QMRF and QPRF reports cannot be generated fully automatically. There are some fields that require user input, e.g., the mechanistic interpretation of the model (if possible) in the QMRF report as required by the fifth OECD Validation Principle. To this end, OpenTox is providing two editors to work with the reports. Figure 5 visualizes that both, the QMRF editor and the QPRF editor, can be used in a flexible way. They can load, edit and store reports to/from the REACH reporting web service (introduced in section 3), as well as to the local file system of the user. Furthermore, it is possible to export reports in PDF format. The following sub-sections here introduce both reporting editors in more detail.

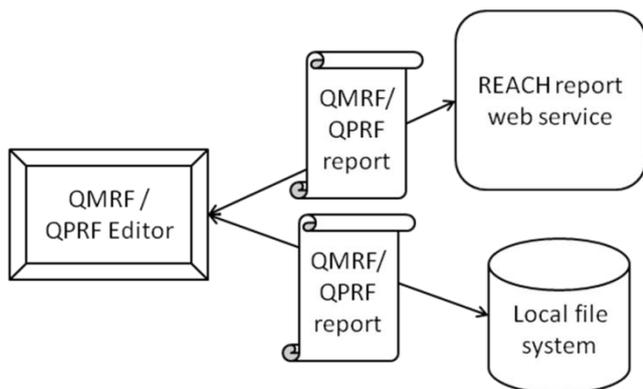


Figure 5: QMRF and QPRF Editors can be used to edit/store/export reports.

4.2.1 QMRF Editor

The original QMRF editor was developed by the OpenTox partner IDEA³¹. It is an open source Java application, and can be started as a Web Start application³². The original functionality allows creating a new report from

³¹ See ambit.acad.bg

³² <http://ambit.sourceforge.net/qmrf/jws/qmrfeditor.jnlp>

scratch. It is further possible to load existing reports that are stored in the predefined QMRF–xml format. Each section of the report can be edited via text fields or forms that provide more guidance (i.e. for QMRF authors). Help dialogs are available for every section. QMRF reports can be stored locally in QMRF–xml format, and can be exported to PDF. This QMRF editor has been extended to meet the new requirements within the OpenTox framework. As described in section 6.1, the QMRF editor will start and directly download the respective QMRF report when adding the suffix '/editor' to the QMRF report URI.³³ It is further possible to manually download another report from the web service. Figure 6 shows a screenshot of the new QMRF editor when manually downloading a report. Moreover, the user can upload changes to the web service, by either overwriting the existing report, or creating a new report on the server.³⁴

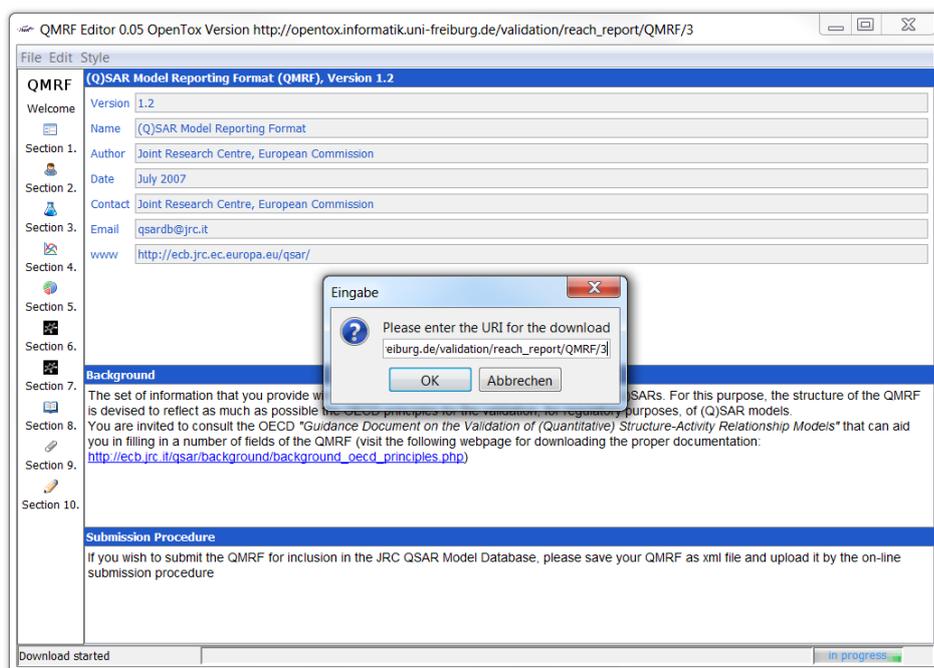


Figure 6: Download a QMRF report with the QMRF editor

4.2.2 QPRF Editor (Q–edit)

Q–edit is a new QPRF editor developed under OpenTox which aims at exploiting implemented web services to provide functionalities that facilitate the creation of QPRF reports by an end user. The editor is designed in a wizard style manner, starting with defining a compound, then entering general information and using a predictive model, and completing with exporting a report as PDF.

³³ Open opentox.informatik.uni-freiburg.de/validation/reach_report/QMRF/3/editor with a Java Web Start–enabled browser.

³⁴ The new version of the QMRF editor supports Authorization & Authentication

Briefly, the main use case consists of the following steps:

- a) Create a new (empty) QPRF report (see Figure 7).

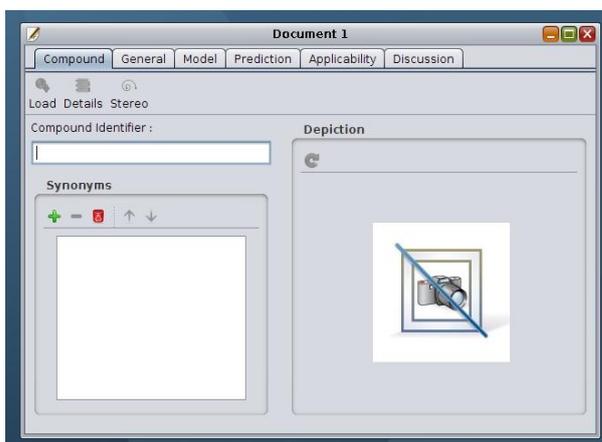


Figure 7: Create a new report

- b) Search for a compound in an on-line database (e.g. AMBIT) (see Figure 8) – Inspect the downloaded compound (View Chemical name(s), SMILES string, CAS RN and a depiction of the compound). Enter additional meta information about the compound, e.g. discuss its stereo-chemical features that might affect the validity of the prediction. (see Figure 9)

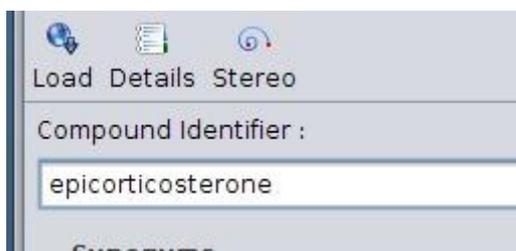


Figure 8: Load a compound from a remote service

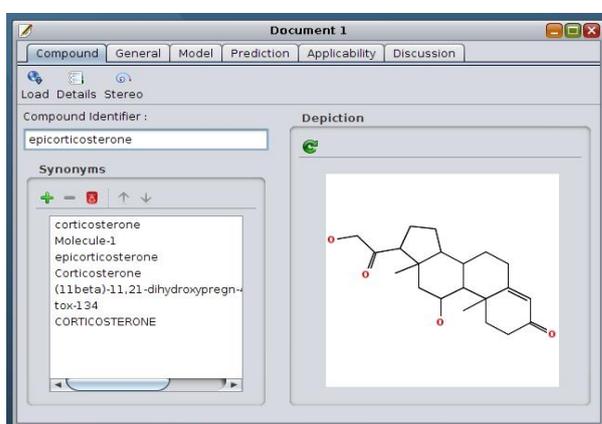


Figure 9: Information loaded from OpenTox web services are presented to the user

The "Details" button gives one access to the various structural attributes of the compound such as its SMILES string as well as to other identifiers including the CAS registration number and the INECS number (see Figure 10).

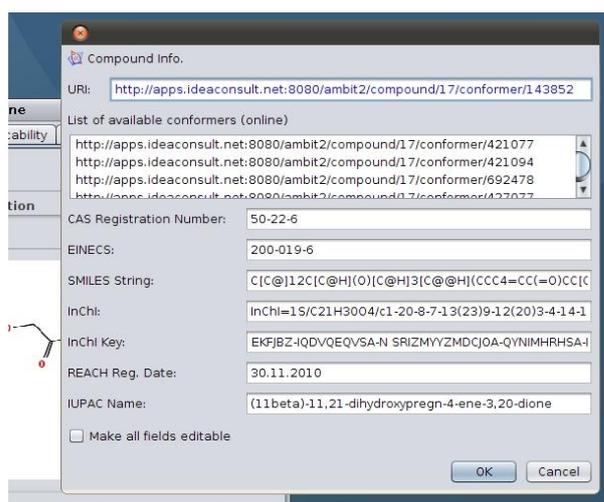


Figure 10: Details about the chemical compound found online

The user is then expected to discuss the stereo-chemical structural attributes of the compound that can possibly affect the reliability of the prediction. The "Stereo" button in the toolbar of the same view will open a new dialog box in which this information should be provided (Figure 11).

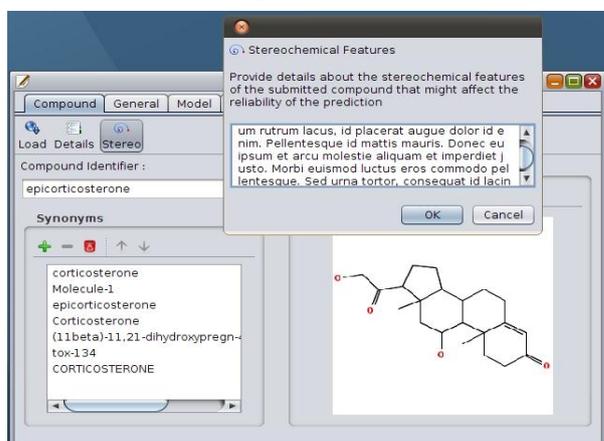


Figure 11: Considerations on stereo-chemical features of the compound

A list of synonyms is also loaded but the user might need to add or remove some synonym that is invalid according to the user's opinion (see Figure 12).

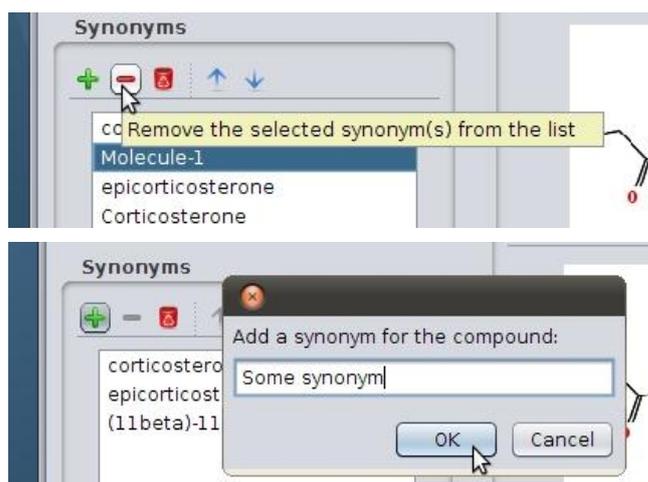


Figure 12: Adding and removing of synonyms

- c) Provide general information about the QPRF report:

The authors of the reports are included using the wizard which can be entered under the second tab.



Figure 13a: Adding an author for the QPRF report

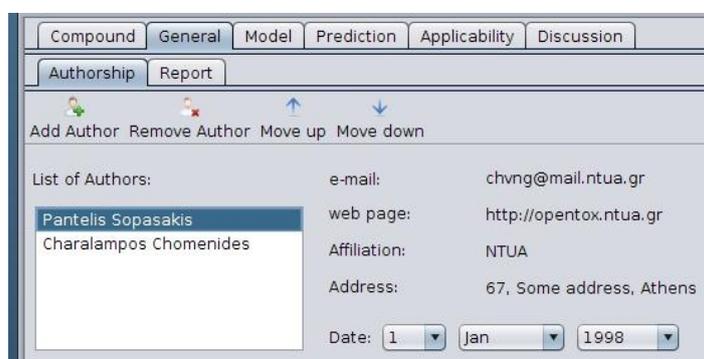


Figure 13b: Entered authors for a QPRF report

- d) Loading of models: models can be loaded using a URI, as listed via <http://opentox.ntua.gr:8080/model> and <http://apps.ideaconsult.net:8080/ambit2/model> and are entered in the respective field in the “Model”-tab (see Figure 14). In case the model is password protected, like with the QMRF Editor, the user has to supply their user credentials or log in as a guest (Figure 15).

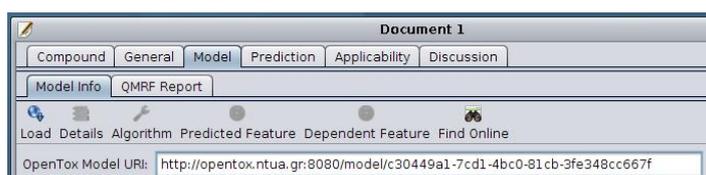


Figure 14: Loading of a model given its URI



Figure 15: Supplying user credentials

Once the model has been loaded the details are displayed, including training dataset features used as well as the parameters used for model construction (see Figures 16 and 17).

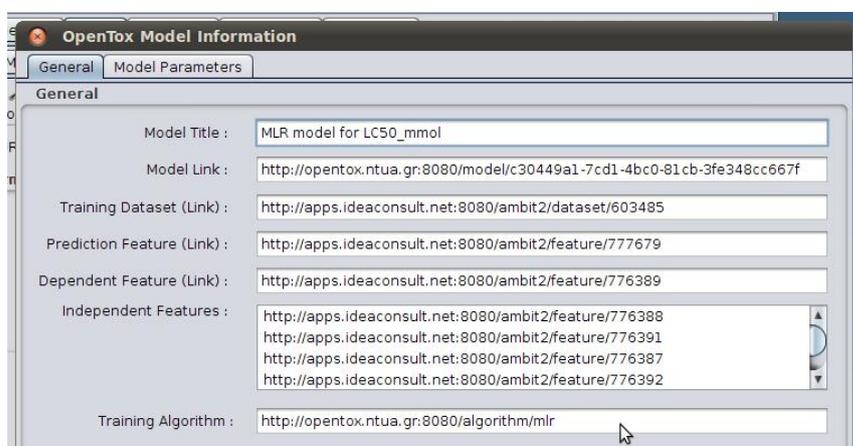


Figure 16: Model details

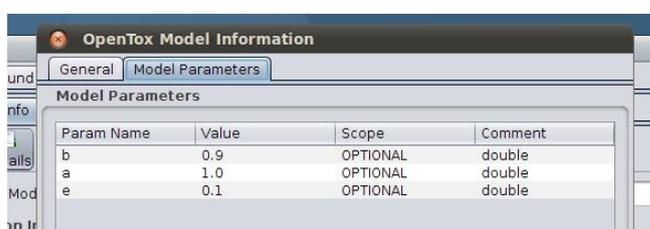


Figure 17: Model parameters

- e) A list of structural analogues can be retrieved on the basis of some similarity index provided by the user. Under the tab "Applicability" one finds the tab "Structural Analogues" where one can provide a similarity threshold in the range 0.5 – 1.0 and click on the button "Acquire List" to get a list of compounds that are similar to the one submitted in the beginning. This is illustrated in the following screenshot where the analogues of sucrose (up to 95% similarity) are listed (see Figure 18)

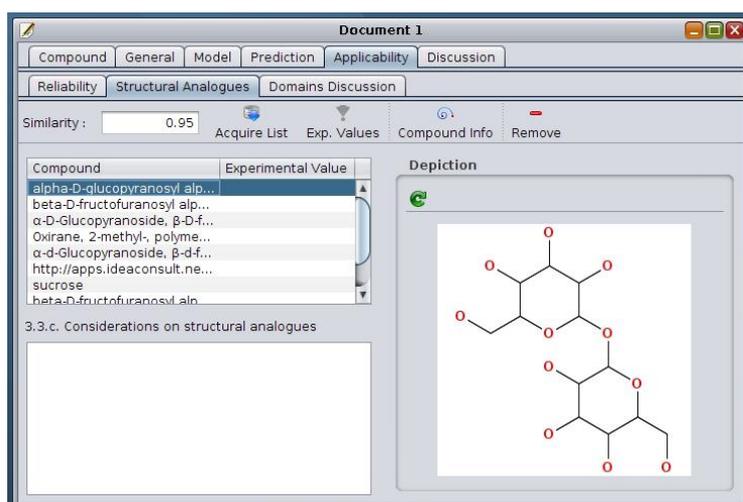


Figure 18: Structural analogues

Once a model is loaded, then one can download all experimental values for each of the structural analogues. Finally, after the list of structural analogues is loaded and one of them is selected, one can inspect its structural information available by clicking on the button "Compound Info" which is found in the toolbar (see Figure 19).

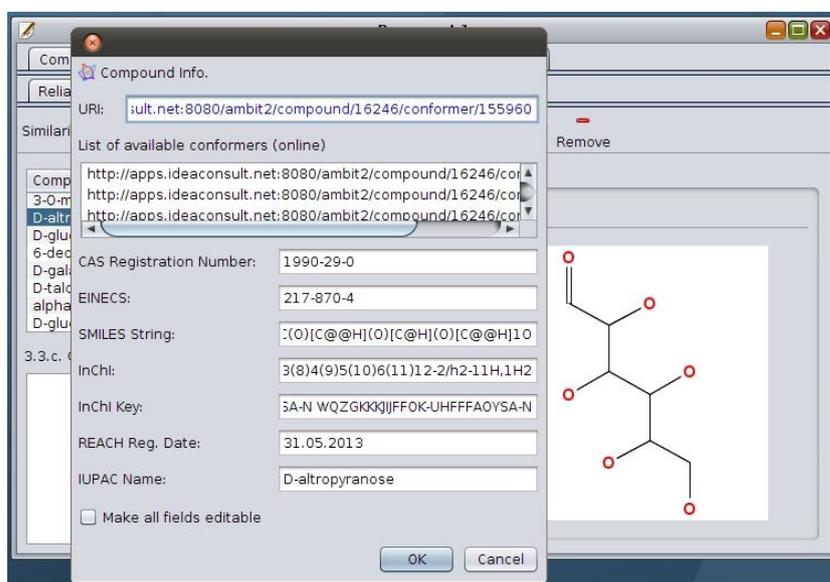


Figure 19: Information about structural analogues

- f) Export the report in PDF format. The resulting document is fully compliant with the standards for QPRF reports that are provided by the EC JRC³⁵.

Users are guided through the above steps with jargon-free documentation that map directly to the sections of the QPRF report as described by the EC JRC. Though it can be used in offline mode, Q-edit is designed to interact with various OpenTox web services providing real-time access to compound databases and model

³⁵ http://hpc.jrc.ec.europa.eu/our_labs/computational_toxicology/qsar_tools/qrf/QPRF_version_1.1.pdf

repositories. QPRF reports are serialized in a compressed binary format so that save/open operations are supported. However, for the sake of uniformity and transparency, QPRF reports are stored in RDF format.

Q-Edit is a tool that allows users to create new prediction reports, and to manage and inspect existing ones. The Q-edit application is written in Java using JDesktop, Swing and AWT and is licensed under the GNU GP License, v.3.0. The source code is available for download from github.com/alphaville/Q-edit and the executable can be downloaded from github.com/alphaville/Q-edit/downloads. It can also be compiled as a Java Web Start application.

5 Validation Routines for Confidential Data

Authentication³⁶ and Authorization³⁷ (abbreviated as A&A) form the core of network security with Accounting being the third 'A' of the trilogy³⁸. **Authentication** is the process of trusting a user's alleged identity by requiring certain evidence such as pairs of id and password or attested digital certificates by some trusted authority. To put it simply, authentication is about confirming that the users are those that they claim to be. **Authorization** is a process that follows authentication and determines access privileges to the system including – but not limited to – retrieval of information from databases and use of web services or other functions of the system. So authorization determines whether a particular authenticated individual has the right to perform a given action and thus frames users with certain restrictions. Finally, **Accounting** refers to the tracking of actions of a particular authenticated user, for example the access and use history of particular services and the consumption of resources such as storage and computational usage.

5.1 REACH legislation and confidential data

As REACH comes into action, thousands of data sheets regarding chemical substances along with safety and exposure information have been registered in a central database run by the European Chemicals Agency (EChA) in Helsinki³⁹. The Agency acts as the central point in the REACH system: it manages the databases necessary to operate the system, co-ordinates the in-depth evaluation of suspicious chemicals and is building up a public database in which consumers and professionals can find hazard information.

According to REACH, the industries are assumed to shoulder the burden of managing the risks of the human contact with chemical substances (in food, cosmetics, etc.) and report to the EU accordingly. EChA publishes information it holds on registered substances free of charge on the Internet. However, in certain cases, information can be withheld, if the registrant submitting the information also submits a justification as to why publishing the information would be potentially harmful to the commercial interests of the registrant or any other party concerned. EChA will not publish the information concerned, if justification is accepted as valid⁴⁰. Towards this direction, REACH-relevant software frameworks such as OpenTox should take into account these confidentiality issues. In the light of these REACH issues, a robust authentication and authorization design is rendered a requirement for the OpenTox framework.

³⁶ en.wikipedia.org/wiki/Authentication

³⁷ en.wikipedia.org/wiki/Authorization

³⁸ en.wikipedia.org/wiki/AAA_protocol

³⁹ echa.europa.eu/

⁴⁰ echa.europa.eu/doc/reachit/dsm_16_confidentiality_claims.pdf

5.2 Authentication and authorization in OpenTox

Within OpenTox, the principles of network security are materialized by means of a central access control system based on Single Sign-On (SSO). Accounting is currently delegated to service providers according to their processing and storage resources. It is fundamental for a distributed system like OpenTox to provide a structured and robust access control system that enables administrators and system providers to:

- Flexibly specify and modify access privileges to users and user groups
- Segregate public and private data
- Protect users' private information such as passwords
- Build web services decoupled from the A&A infrastructure (administrative access to some database may not be necessary) or even provide completely public services without A&A

For these reasons, SSO was chosen as the security mechanism in OpenTox. The principles of SSO and how these bind with REST and OpenTox web services, was previously described in detail in the [OpenTox Report on Tools for Access to Confidential Information](#)⁴¹. The REST API for accessing the SSO infrastructure is described in the OpenTox API 1.2 at <http://www.opentox.org/dev/apis/api-1.2/AA>

5.2.1 Topological description of access control

The realization of access control in OpenTox is currently based on a central SSO server which is employed by individual web services to decide on a user's access to them or to other services to which the former act as gateways or proxies. Figure 20 depicts the main concept and how services interact with the single access control manager when a single service is involved.

The client identifies itself providing an authentication token⁴² to the OpenTox web service it wants to access. Tokens are generated by the SSO services upon request (over a secure TLS-encrypted connection⁴³, i.e. a connection using the Transport Layer Security protocol as described by the RFC-5246⁴⁴ specifications) of the user's identifier and password (user credentials) and have a certain lifetime. In the current implementation, tokens stay active for 24 hours unless they are invalidated by the client. The web service receives this token, and using the SSO service, checks whether the token is valid (corresponds to a logged in user) and whether that user is granted the necessary privileges to perform the request. If authentication or authorization fails, a status code 401⁴⁵ is returned to the user along with an error report⁴⁶.

⁴¹ <http://www.opentox.org/data/documents/development/opentoxreports/opentoxreportd33/view?searchterm=D3.3>

⁴² en.wikipedia.org/wiki/Security_token

⁴³ en.wikipedia.org/wiki/Transport_Layer_Security

⁴⁴ tools.ietf.org/html/rfc5246

⁴⁵ [HTTP Status code 401 definition: www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2)

⁴⁶ [OpenTox specifications for Asynchronous Tasks and Error Reports: http://opentox.org/dev/apis/api-1.2/AsyncTask](http://opentox.org/dev/apis/api-1.2/AsyncTask)

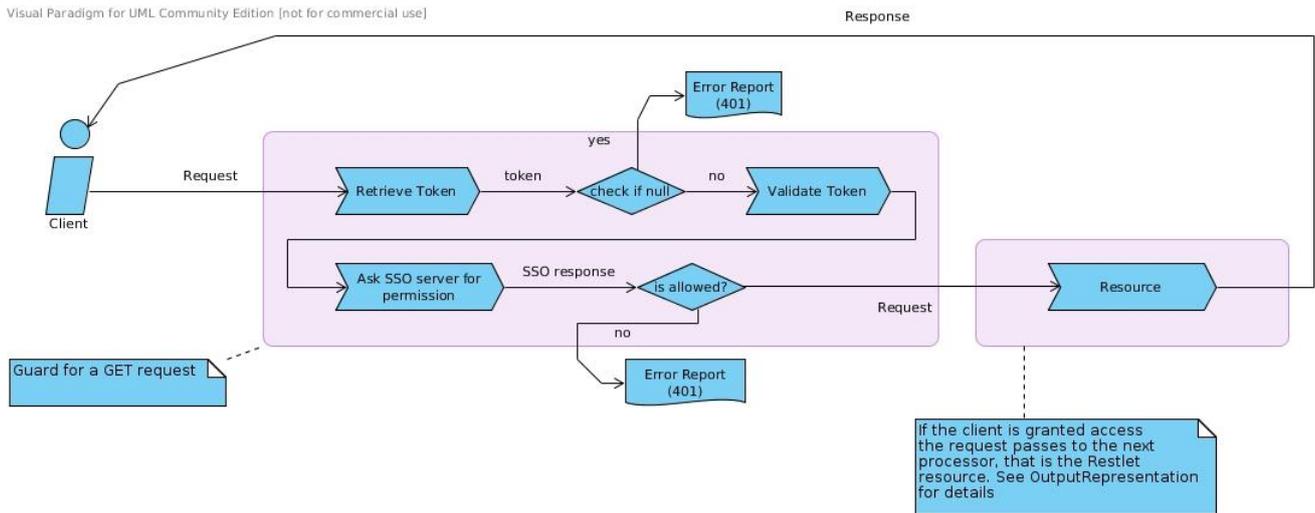


Figure 20: Protection of confidential information in the request-response chain

In case the initial client request induces a second request from the invoked service, this is always done on behalf of the user using the provided token. This token is passed to the next service(s) of the workflow and in case authorization fails somewhere in the middle, an error report is generated and propagated backwards to the client with a status code 401⁴⁷. In the scheme described in Figure 21, service 1 passes to the remote service the token of the user that initiated the request. In this way, it is guaranteed that an end user will not access either directly or indirectly (through some other service) confidential data, unless he is authorized to do so.

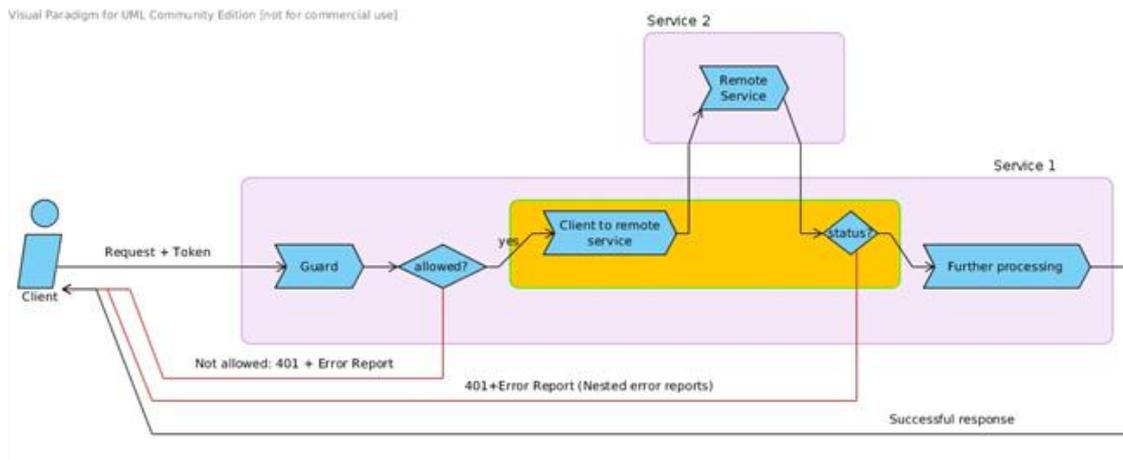


Figure 21: Protection of confidential data in a multi-service application

5.2.2 Managing access

Access to confidential data is secured by the SSO service. The way in which this service allows or blocks an action on an OpenTox web service is specified by the **policy** for the underlying resource. A policy over a

⁴⁷ HTTP Status code 401 – Unauthorized: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2>

resource (identified by its URI) defines to whom access is granted (Figure 22). The authorization policies for the central SSO server are defined by the creator of each resource. SSO policies specify restrictions on the REST level⁴⁸ and with respect to some HTTP method. These restrictions apply either on individual users or on groups.

From a programmatic point of view, a policy here is implemented as an XML file specifying explicitly to whom access is allowed and under which conditions. This way a policy defines rules that specify who or what can access these protected resources. The rules are, in effect, permissions describing when and how a user can perform an action on a given protected resource. A user can be an individual or a group. In general, the permissions define what a user can do to which resource and under what conditions.

For OpenTox, we provide a Policy Configuration Service (PCS) to define such preferences and manage the policies. The service allows any registered user to define, modify, and revoke permissions on specific resources (URIs). After creation, only the resource owner (the user who created it) can alter the policy.

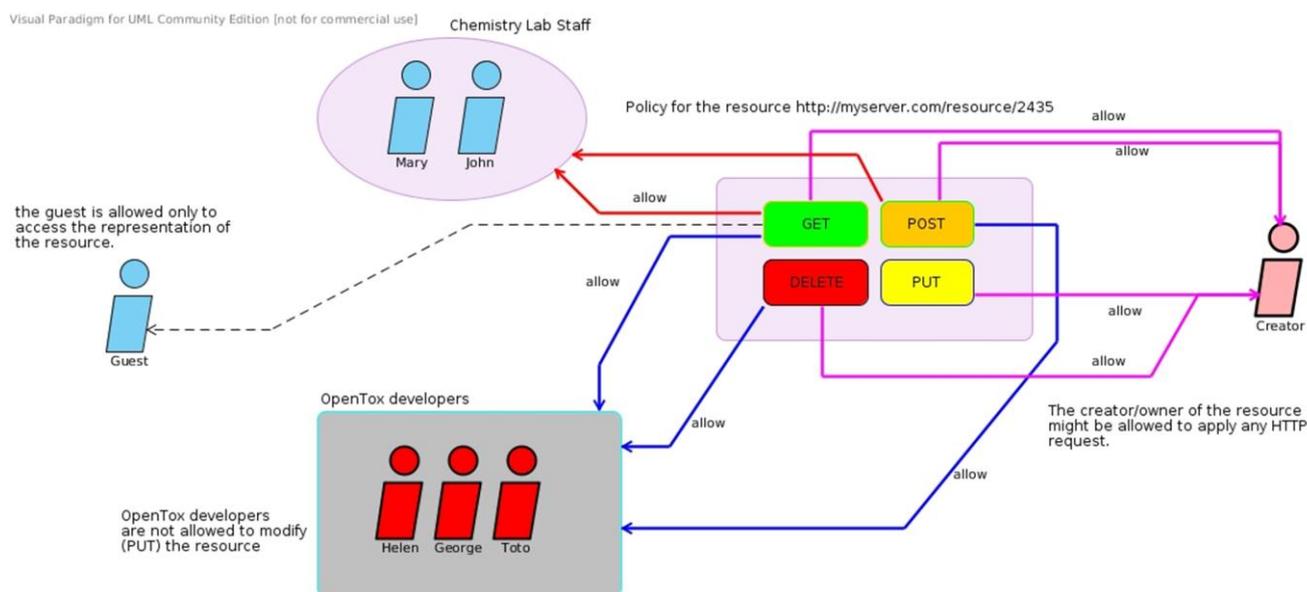


Figure 22: Policy definition – A flexible way to assign privileges to individual users and groups

The policy for a new resource, is created by the owner of the resource, which is the individual that generates it using some web service. For example when a user uploads a new dataset to a dataset server (using POST) the PCS also creates a policy for it. The policy is created indirectly as the service that accepts the client's request also creates the policy for it. Finally, we provide an example policy XML which is POSTed to the SSO policy service to define access rules for the hypothetical resource <http://opentox.org/s2> in Figure 23.

⁴⁸ For a short explanation and reference to the REST commands, please see: www.opentox.org/dev/framework/restweb

```

<?xml version="1.0" encoding="UTF-8"?>
<Policies>
<Policy name="s2_policy" createdby="id=amadmin,ou=user,dc=opensso,dc=java,dc=net"
lastmodifiedby="id=amadmin,ou=user,dc=opensso,dc=java,dc=net"
creationdate="1275290803394"2">
  <Rule name="s2 rule lastmodifieddate="1275290803394"
    <ServiceName name="iPlanetAMWebAgentService"/>
    <ResourceName name="http://opentox.org/s2"/>
    <AttributeValuePair>
      <Attribute name="POST"/>
      <Value>allow</Value>
    </AttributeValuePair>
    <AttributeValuePair>
      <Attribute name="GET"/>
      <Value>allow</Value>
    </AttributeValuePair>
  </Rule>
  <Subjects name="s2 subject 2" description="">
    <Subject name="amaunz" type="LDAPUsers" includeType="inclusive">
      <AttributeValuePair>
        <Attribute name="Values"/>
        <Value>uid=amaunz,ou=people,dc=opentox,dc=org</Value>
      </AttributeValuePair>
    </Subject>
  </Subjects>
</Policy></Policies>
    
```

Figure 23: A policy in XML format

5.2.3 Policy Creation and management

When a client creates a resource, it should be able to specify a policy for it by passing some parameters to the corresponding service. This can be done for simplicity using POST parameters like "policy=public" or "allow_users_get=john,nick", "allow_users_post=nick" or "allow_groups_get=development,partner" etc. The client should be able to specify a policy by providing an XML document for it. To avoid passing the policy as a form parameter (in MIME-type application/x-form-urlencoded) a Header parameter can be used instead:

```
Policy = "Policy: <XML for policy>"
```

In Figure 24 the way policies are created is presented for the use case of model creation. The user that initiates the training will either provide a policy XML on the header of the request or the policy definition is delegated to the trainer. The default policy for models defines that only the creator is allowed to perform predictions and delete the model while the RDF representation of the model is publicly available. Once the policy for the model is created, only the creator is allowed to modify it and grant specific access to other users and/or groups.

Visual Paradigm for UML Community Edition [not for commercial use]

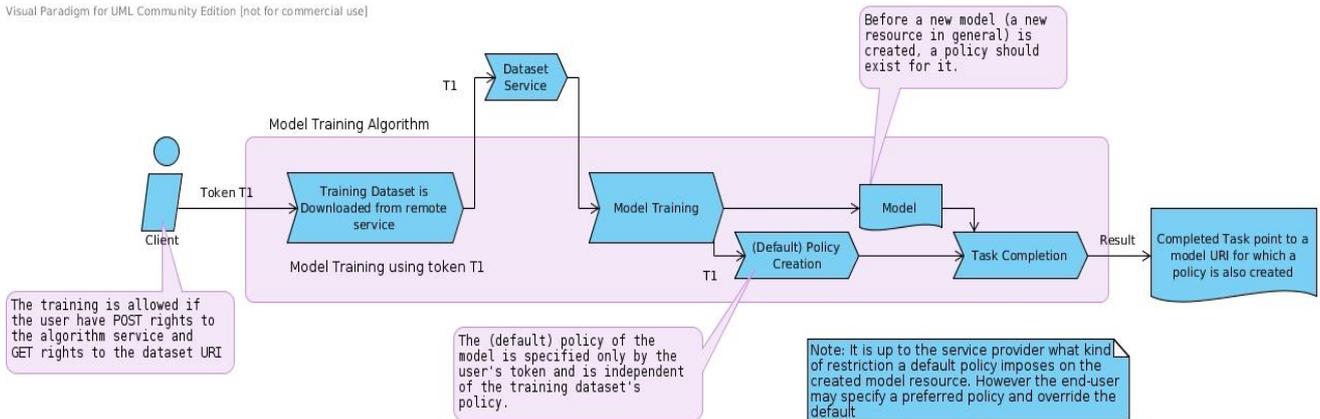


Figure 24: Policy creation in model training

5.2.4 Evaluation of the current access control system

The following evaluation not only provides an insight, from the web service developer point of view, regarding the access control system of OpenTox but also justifies the *raison d'être* for it. The adoption of SSO as an access control system for OpenTox offers the following features:

4. The web services are designed, implemented and deployed without the need for the maintenance of a local users' database. No administrators or privileged users are needed to deploy an OpenTox web service thus underlining the open nature of the framework since everyone can design and deploy an OpenTox-based service. Therefore, the web services are disengaged from the authentication and authorization infrastructure (A&AI). Phishing⁴⁹ opportunities are reduced to a minimum since users provide their credentials only once for every session.
5. Reduces password fatigue⁵⁰ as users are not required to remember as many pairs of username and password as the OpenTox web services they need to access. It also reduces the time that the user spends in entering passwords.
6. The client authenticates against the SSO service establishing an encrypted SSL/TLS connection and using it to pass the pair of username and password. No credentials are transferred over unencrypted connections and, more, these credentials are not passed to the individual services. The SSO service verifies the received credentials against the opentox.org's user database (LDAP⁵¹) which is not exposed to the network whatsoever.
7. The creator of a resource is responsible for its availability (public, private, etc). The data held by a web service and their flow to third party users or groups of such is fully controlled by the creator.

What is considered to be a possible drawback of this design approach is that the SSO server is the most critical node in the system. An outage of the SSO service will affect all services that depend on it.

⁴⁹ Definition of phishing: en.wikipedia.org/wiki/Phishing

⁵⁰ Definition of password fatigue: encyclopedia.thefreedictionary.com/password+fatigue

⁵¹ Project page of openLDAP: www.openldap.org/

5.2.5 OECD Principles

Validation services, even when running against confidential data, should satisfy the 5 OECD principles for (Q)SAR validation. In particular, OpenTox validation web services comply with OECD principles 3 and 4, even when confidential data are involved and A&A services are activated:

PRINCIPLE 3: "DEFINED APPLICABILITY DOMAIN"

OpenTox provides tools for the determination of applicability domains during the validation of (Q)SAR models against confidential datasets.

PRINCIPLE 4: "APPROPRIATE MEASURES OF GOODNESS-OF-FIT, ROBUSTNESS AND PREDICTIVITY"

OpenTox provides scientifically sound validation routines for the determination of these measures.

5.3 Use Case description

The problem arises when different or seemingly conflicting access privileges are expected to occur regarding models, datasets and other services. A client needs to validate a model against confidential data to which he might have no access. Regarding user privileges, the following alternative cases may occur:

	Access to the test dataset	
Access to the QSAR model	Yes/Yes	Yes/ No
	No /Yes	No /No

The most representative cases are the Y/Y and the N/N case (as the Y/N and N/Y cases are actually sub-cases of the N/N case). In case that the user has access both to the test dataset and the QSAR model (Y/Y), the framework has to take care of the access privileges on any resources (datasets) created as predictions from the model so that confidential information will not leak from the validation service. Current access control infrastructure of OpenTox, combined with the REST architecture, caters for the protection of all these resources. In the second case, where the user has not access either to the model or to the test dataset, it becomes evident that a second user with enhanced privileges has to intervene and perform the validation on behalf of the first user exposing back to him just the validation report but no information regarding the test set and/or the model. For validation purposes OpenTox can provide a facility to test (Q)SAR models remotely against confidential datasets without getting access to the actual entries of the database to ensure security and confidentiality of proprietary data.

5.3.1 Use Case implementation

Current OpenTox implementations support the case where the end user provides his own dataset or has access to the confidential dataset. When confidential data are held by public servers and these are to be used in a validation session, it should be clear which new resources that are created replicate some part of these data and under what kind of policies these resources are created. Validation lies in between all other OpenTox services and creates models and datasets on behalf of the end user. In Figure 25 the validation procedure is described regarding the service invocations involved.

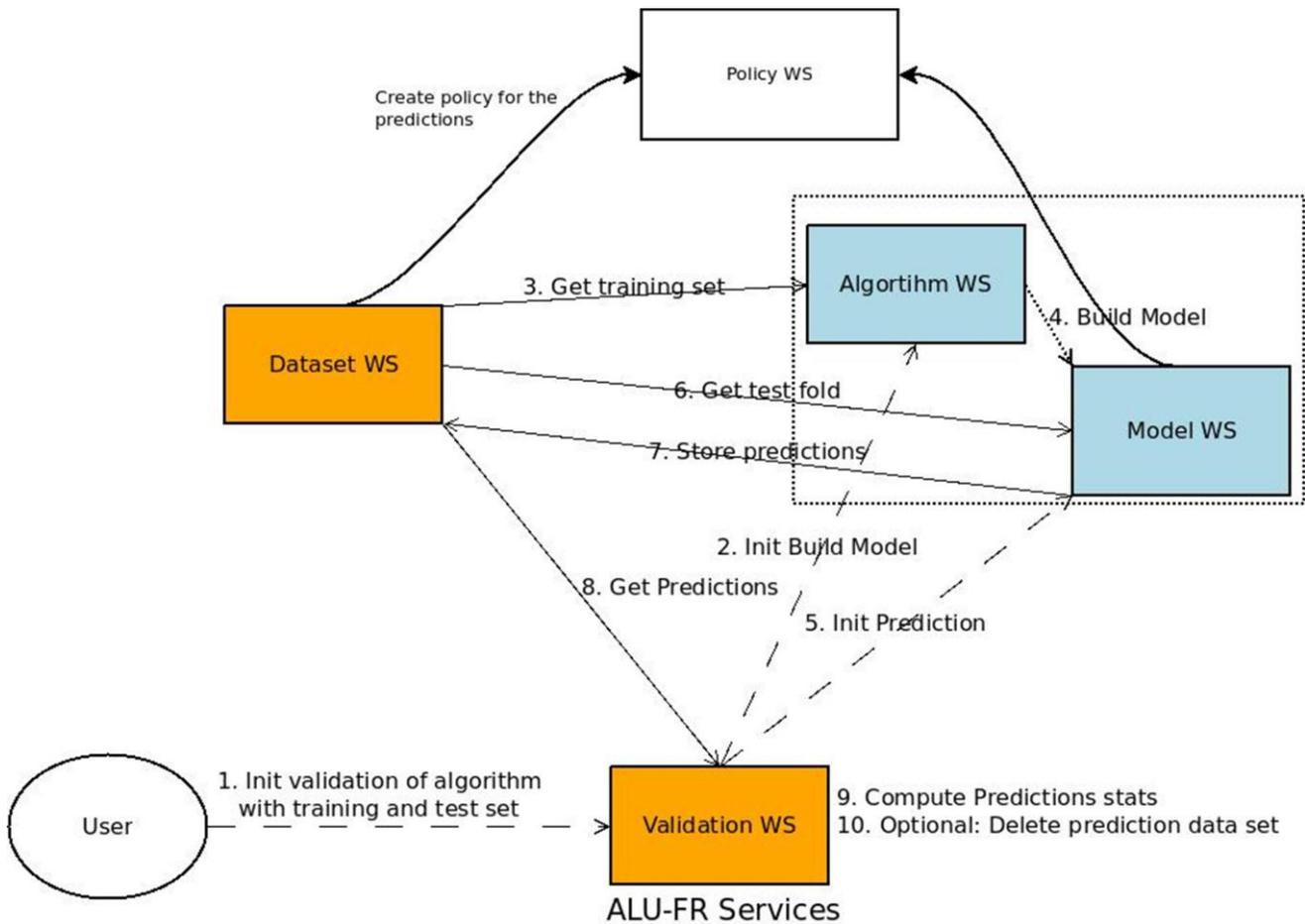


Figure 25: Topological description of an OpenTox-compliant validation service (interactions with other web services)

All service invocations mentioned above (validation request, model training, predictions) are performed using the end user's authentication token. In case a new resource is to be created, as for example in the case a model is trained or a dataset with predicted values is created on a dataset service, a policy is defined by the corresponding service that creates the resource (using again the user's token) and is POSTed to the policy service. All created models and datasets with predictions "belong" to the user that initiated the validation and only that user can amend their access options.

6 Validation examples using confidential data

Two different examples are presented in this section. The first example shows the seamless integration of the A&A concept into the OpenTox application ToxCreate. The user does not have to worry about security issues while he benefits from the comfort of a Graphical User Interface (GUI). The second example gives more technical insights: a remote confidential dataset is validated using the command line tool cURL⁵². This example emphasizes how confidentiality is guaranteed with locally distributed web services.

⁵² URL is a command-line tool serving as an HTTP client. See curl.haxx.se

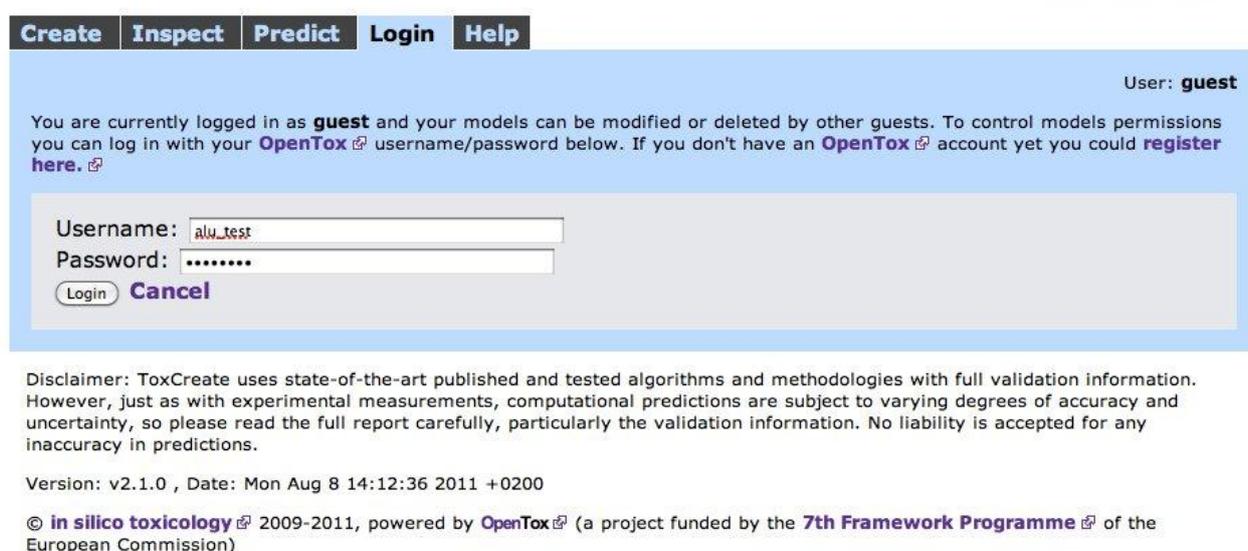
6.1 Validation against confidential data with ToxCreate

ToxCreate⁵³ is a web-based application developed within the OpenTox framework. It is based on various OpenTox web services and provides model creation, validation and the prediction of compounds with the created models⁵⁴. The user can use already created models, or upload a new dataset to train a new model. This example focuses on the latter use case: assuming that the data provided by the user is confidential, no other user should be able to access the uploaded dataset or resources created on the basis of this dataset, unless the creator provides an override to this default to specific users.

In ToxCreate the user is automatically logged in as guest. The application is organized with multiple tabs (see Figure 26). The user can login at the 'Login' tab. This demonstration is performed with the test-user 'alu_test' (password is 'alu_test' as well).

Creates computational models to predict toxicity

ToxCreate



User: **guest**

You are currently logged in as **guest** and your models can be modified or deleted by other guests. To control models permissions you can log in with your **OpenTox**  username/password below. If you don't have an **OpenTox**  account yet you could **register here**. 

Username:

Password:

Disclaimer: ToxCreate uses state-of-the-art published and tested algorithms and methodologies with full validation information. However, just as with experimental measurements, computational predictions are subject to varying degrees of accuracy and uncertainty, so please read the full report carefully, particularly the validation information. No liability is accepted for any inaccuracy in predictions.

Version: v2.1.0 , Date: Mon Aug 8 14:12:36 2011 +0200

© **in silico toxicology**  2009-2011, powered by **OpenTox**  (a project funded by the **7th Framework Programme**  of the European Commission)

Figure 26: Login screen of ToxCreate

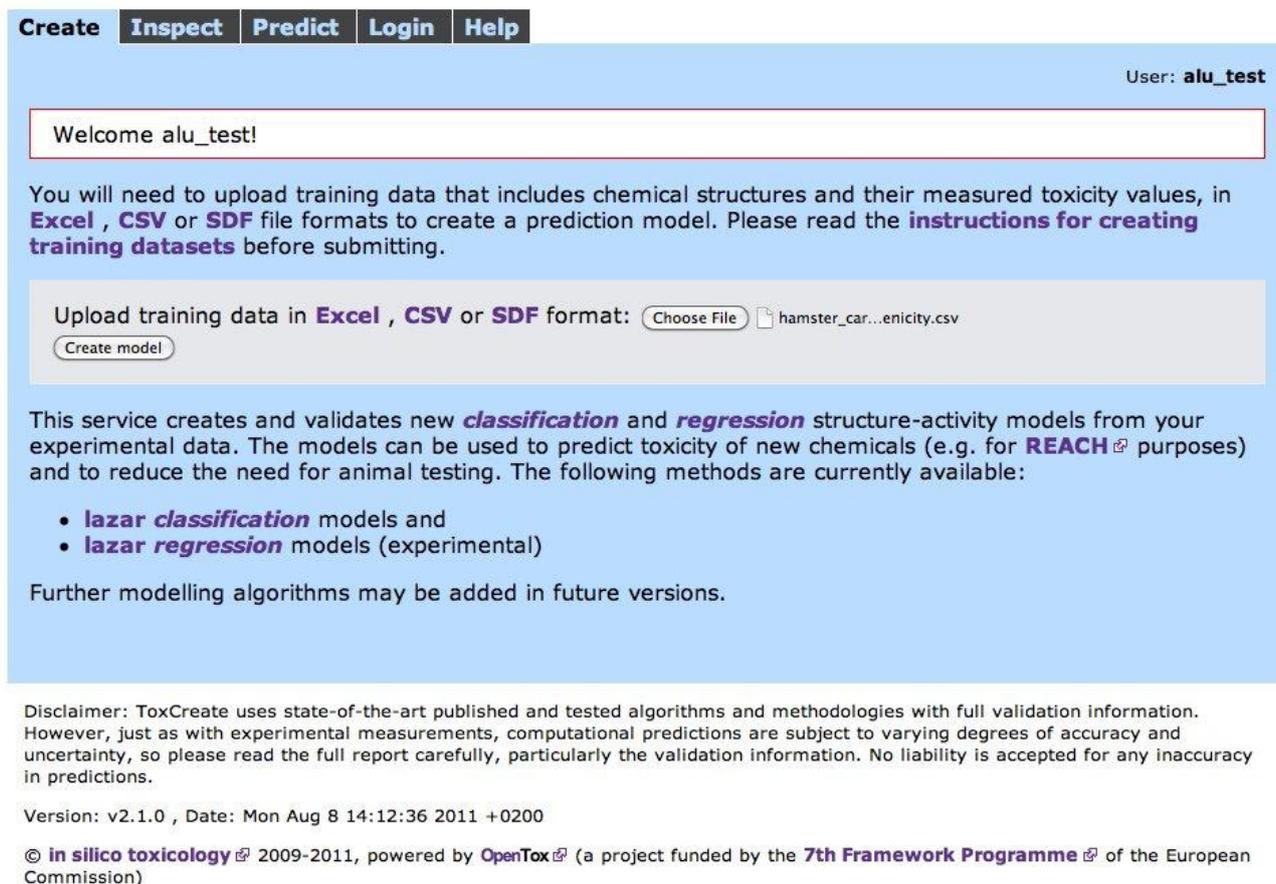
After logging in (Figure 27), the current user switches from 'guest' to 'alu_test', as shown on the top right of the web page. It is now possible to safely upload the confidential dataset. For this example a publicly available dataset from the Carcinogenic Potency Database (CPDB) was chosen: the hamster carcinogenicity dataset contains 85 compounds⁵⁵. A binary target variable indicates whether the compound is active or inactive. This dataset can be uploaded from your local hard drive at the 'Create' tab of ToxCreate (Figure 27).

⁵³ The latest production version of ToxCreate running at www.toxcreate.org

⁵⁴ For more info on ToxCreate see <http://opentox.net/dev/testing/testcasedevelopment/toxcreate>

⁵⁵ Available at https://github.com/helma/opentox-test/blob/master/data/hamster_carcinogenicity.csv

Creates computational models to predict toxicity

[Create](#) [Inspect](#) [Predict](#) [Login](#) [Help](#)
User: **alu_test**

Welcome alu_test!

You will need to upload training data that includes chemical structures and their measured toxicity values, in **Excel** , **CSV** or **SDF** file formats to create a prediction model. Please read the **instructions for creating training datasets** before submitting.

Upload training data in **Excel** , **CSV** or **SDF** format: hamster_car...enicity.csv

This service creates and validates new **classification** and **regression** structure-activity models from your experimental data. The models can be used to predict toxicity of new chemicals (e.g. for **REACH** purposes) and to reduce the need for animal testing. The following methods are currently available:

- **lazar classification** models and
- **lazar regression** models (experimental)

Further modelling algorithms may be added in future versions.

Disclaimer: ToxCreate uses state-of-the-art published and tested algorithms and methodologies with full validation information. However, just as with experimental measurements, computational predictions are subject to varying degrees of accuracy and uncertainty, so please read the full report carefully, particularly the validation information. No liability is accepted for any inaccuracy in predictions.

Version: v2.1.0 , Date: Mon Aug 8 14:12:36 2011 +0200

© **in silico toxicology** 2009-2011, powered by **OpenTox** (a project funded by the **7th Framework Programme** of the European Commission)

Figure 27: User interface provided by ToxCreate for uploading a training dataset

When the 'Create model' button is pressed, ToxCreate automatically switches to the 'Inspect' tab. In the background the dataset is uploaded and a model building and validation process is initialized:

- The dataset is uploaded to the dataset web service, and registered at the A&A server to allow access to user 'alu_test' (and the group of this user) only.
- Structural features are mined on this dataset and a lazar model is built. This model can be used later on to make predictions ('Predict' tab).
- A 10-fold cross-validation to evaluate the predictive power of this model on the dataset is performed. This splits the datasets into 10 folds, and repeatedly builds a model on 9 different folds of the 10 dataset folds. The resulting model is used to predict the test dataset (the fold that was left out when building the model). The final results of this cross-validation are shown in the validation section of the model's properties. More details are available in the validation report.
- Finally a QMRF report is automatically created for this model. It contains meta-information on the trained model and the algorithm, the validation results, and other information about the model.

The results of these steps are gradually added to the new 'Hamster Carcinogenicity' model that is available on the 'Inspect' tab, until the status is finally set to completed (Figure 28).

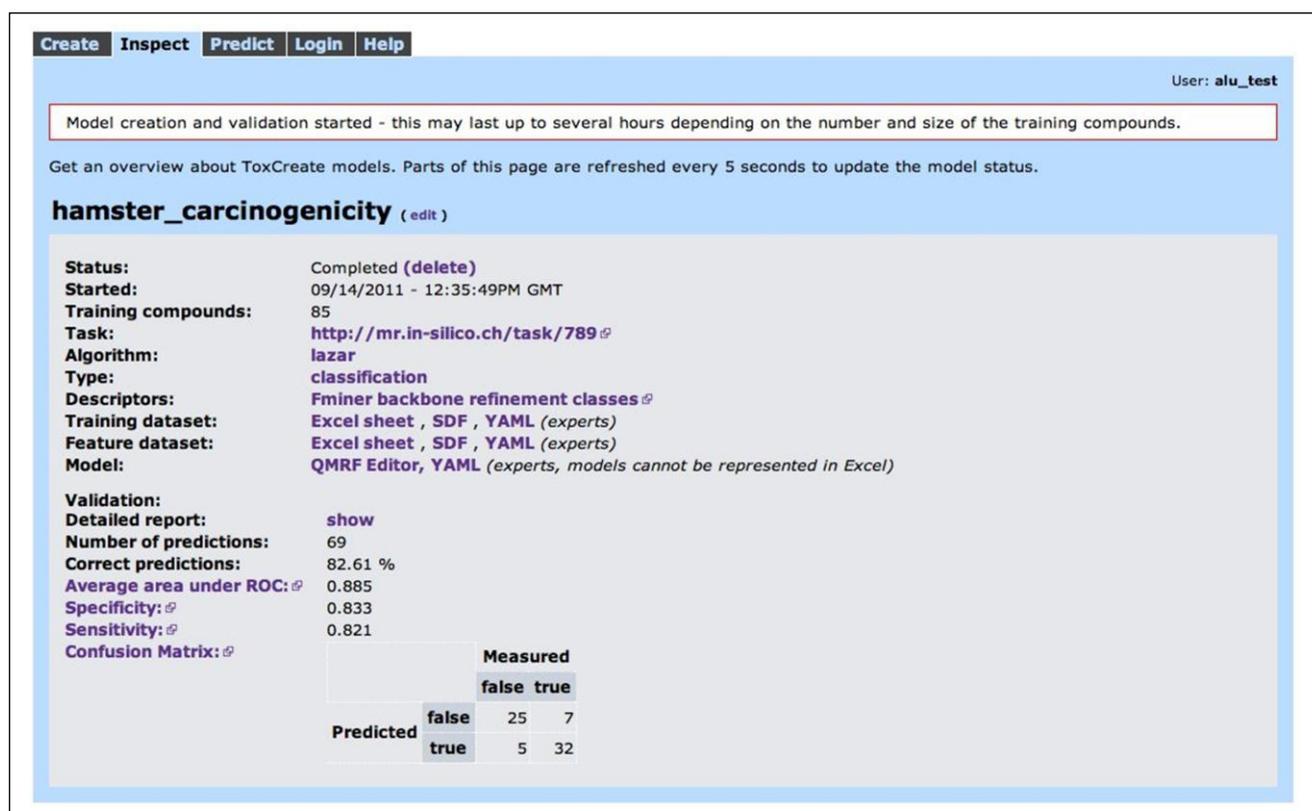


Figure 28: An overview of a QSAR model produced within the ToxCreat application

The user can now have a look at the detailed validation report, edit the QMRF report with the QMRF editor, or make predictions with the newly created model.

Note that the resources (dataset, model, validation report, etc...) are only available to user 'alu_test'. After logging out, the user 'guest' has no access to the newly created Model 'Hamster Carcinogenicity'; it is not available in the 'Inspect' tab of ToxCreat.

6.2 Validation against validation data using distributed web services

This example demonstrates how a dataset is protected by A&A and SSL when used for validation. The use case is a training-test-split validation. This is an established method to estimate the performance of a prediction model on unseen data⁵⁶: the original dataset is split into a training data set and a test data set. The training data set is used to build a model. The model is then applied to make predictions on the unseen test dataset.

In the use case presented here, we use the well-known Caco-2 dataset⁵⁷. The dataset consists of 100 organic molecules with a numeric endpoint (Caco-2 permeability, $\log P_{app}$). The dataset was uploaded to the AMBIT2 dataset service (It is available with A&A at <https://ambit.uni-plovdiv.bg:8443/ambit2/dataset/R401560>). 27

⁵⁶ More extensive techniques like cross-validation should be preferred especially if the training dataset is small. The simpler training test split method is chosen for proof of concept.

⁵⁷ pubs.acs.org/doi/suppl/10.1021/ci049884m

numerical features have been calculated using AMBITs descriptor calculation services⁵⁸. A linear regression algorithm (<https://ambit.uni-plovdiv.bg:8443/ambit2/algorithm/LR>) was selected to predict the target variable. Both, the dataset as well the algorithm service are located in Sofia, Bulgaria. The validation server is located at the University of Freiburg, Germany.

We are executing this example with the command line tool curl (<http://curl.haxx.se/>), using the functionality specified in the OpenTox API⁵⁹. Alternatively, the REST calls could be performed with any programming language that includes a REST library. To this end, the validation routines can be integrated into an application with a GUI (like in the ToxCreate example above).

6.2.1 Login:

The first step is to derive a subject-id from the SSO-server, for the user 'guest': ⁶⁰

The curl call returns the following:

```
curl -X POST -d "username=guest" -d "password=guest" http://opensso.in-silico.ch/opensso/identity/authenticate?uri=service=openldap
```

```
token.id= AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#
```

This string can now be used to identify the client as user 'guest' in the subsequent cURL calls. The access to all resources that are created with this subject-id will only be granted to user 'guest' (and the group of this user).

6.2.2 Start validation:

The validation is initialized with a HTTP POST call to http://opentox.informatik.uni-freiburg.de/validation/training_test_split. The parameters to control the routine are algorithm-URI, dataset-URI and prediction-feature. The subject-id is specified as additional header (with -H option):

```
curl -X POST -d algorithm_uri="https://ambit.uni-plovdiv.bg:8443/ambit2/algorithm/LR" -d dataset_uri="https://ambit.uni-plovdiv.bg:8443/ambit2/dataset/R401560" -d prediction_feature="https://ambit.uni-plovdiv.bg:8443/ambit2/feature/22190" http://opentox.informatik.uni-freiburg.de/validation/training_test_split -H "subjectid:AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#"
```

```
http://opentox.informatik.uni-freiburg.de/task/1004
```

The validation service returns the URI of a task object, while running the validation as an asynchronous background job. The result is stored in the task object when the job is finished.

Get validation result-URI:

⁵⁸ See for example: <https://ambit.uni-plovdiv.bg:8443/ambit2/algorithm/org.openscience.cdk.qsar.descriptors.molecular.AtomCountDescriptor>

⁵⁹ Documentation for the OpenTox API <http://opentox.org/dev/apis/api-1.2>

⁶⁰ The cURL calls (presented in purple boxes) can be copied to and executed with a command-line interface. The return value is marked in green (success) or orange (error) boxes.

Task resources are not protected, which is why the following cURL call does not need to include the subject-id:

```
curl http://opentox.informatik.uni-freiburg.de/task/1004 -H "Accept:application/x-yaml"
```

```
---
http://purl.org/dc/elements/1.1/title: Perform training test split validation
http://www.opentox.org/api/1.1#hasStatus: Completed
http://www.opentox.org/api/1.1#resultURI: http://opentox.informatik.uni-freiburg.de/validation/114
http://www.opentox.org/api/1.1#percentageCompleted: 100.0
[...]
```

Using cURL on the task-URI returns a list of its properties, including the field result-URI that contains the validation-URI.

Get validation result:

The following cURL call demonstrates that the validation result <http://opentox.informatik.uni-freiburg.de/validation/114> is protected by the A&A routines:

```
curl http://opentox.informatik.uni-freiburg.de/validation/114
```

```
--- !ruby/object:OpenTox::ErrorReport
actor: http://opentox.informatik.uni-freiburg.de/validation/114
errorType: OpenTox::NotAuthorizedError
http_code: 401
message: Not authorized
[...]
```

Access is denied, and an error report is returned instead. However, we can access this resource when specifying the subject-id:

```
curl http://opentox.informatik.uni-freiburg.de/validation/114 -H
"subjectid:AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#"
```

```

---
http://www.opentox.org/api/1.1#model: https://ambit.uni-
plovdiv.bg:8443/ambit2/model/35009
http://www.opentox.org/api/1.1#trainingDataset: http://opentox.informatik.uni-
freiburg.de/dataset/452
http://www.opentox.org/api/1.1#predictionDataset: http://opentox.informatik.uni-
freiburg.de/dataset/454
http://www.opentox.org/api/1.1#predictionFeature: https://ambit.uni-
plovdiv.bg:8443/ambit2/feature/22190
http://www.opentox.org/api/1.1#numInstances: 32
http://www.opentox.org/api/1.1#testTargetDataset: https://ambit.uni-
plovdiv.bg:8443/ambit2/dataset/R401560
http://www.opentox.org/api/1.1#validationType: training_test_split
http://www.opentox.org/api/1.1#testDataset: http://opentox.informatik.uni-
freiburg.de/dataset/453
http://www.opentox.org/api/1.1#algorithm: https://ambit.uni-
plovdiv.bg:8443/ambit2/algorithm/LR
http://www.opentox.org/api/1.1#regressionStatistics:
  http://www.opentox.org/api/1.1#rootMeanSquaredError: 0.627121310539419
  http://www.opentox.org/api/1.1#rSquare: 0.352408295243186

```

The validation object links to resources that have been used for validation, e.g., training and test data sets. The latter have been created by splitting the original dataset, and are located at Freiburg's dataset service.

Create report:

We finally create a validation report from the validation resource:

```

curl -X POST -d validation_uris="http://opentox.informatik.uni-
freiburg.de/validation/114" http://opentox.informatik.uni-
freiburg.de/validation/report/validation -H
"subjectid:AQIC5wM2LY4SfczngIclWu3ztAWK7WKXHfAFK+CI8Rvf5zU=@AAJTSQACMDE=#"

```

Again this call returns a task object first (skipped for simplicity). Accessing this task reveals the report-URI:

```

---
http://www.opentox.org/api/1.1#resultURI: http://opentox.informatik.uni-
freiburg.de/validation/report/validation/14
[...]

```

Visit validation report with browser:

The validation report could be requested with cURL as well, but is best viewed with a web browser. As it is protected by A&A, access is denied for <http://opentox.informatik.uni-freiburg.de/validation/report/validation/14> (Figure 29).

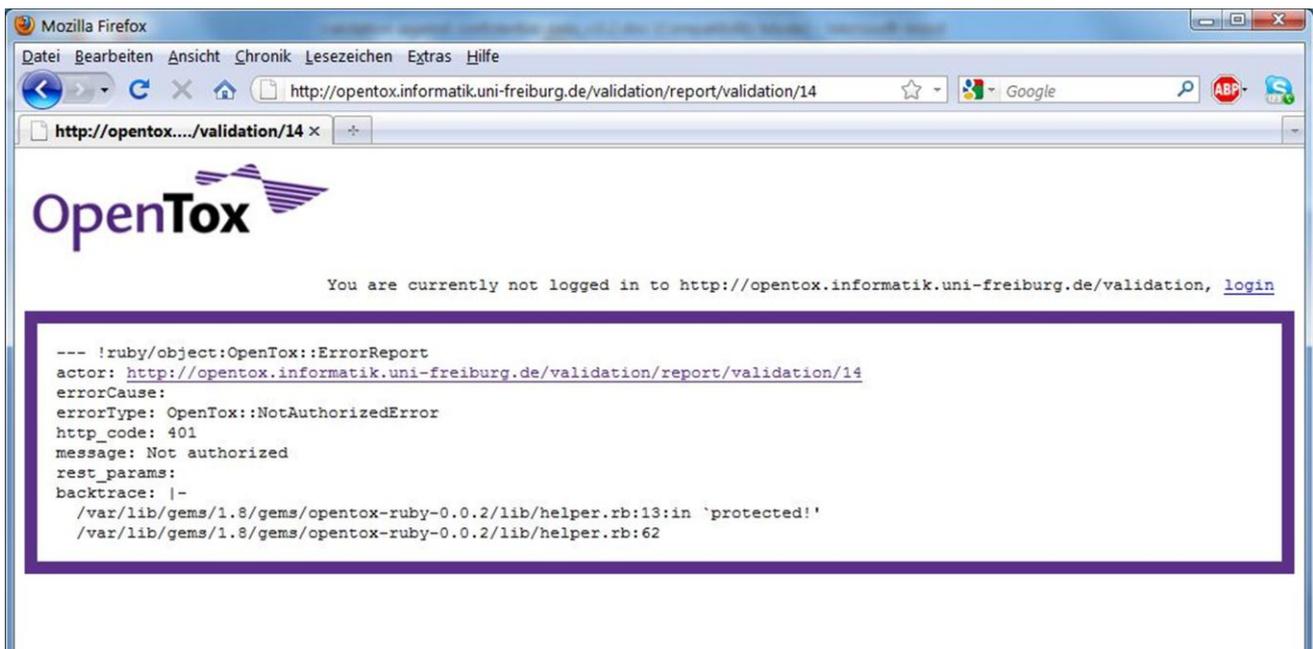


Figure 29: Protected resources do not allow unauthorized access – access has been denied to a validation report

The user has to login as guest⁶¹ (using the link at the top right of the browser, password is 'guest'). This stores the subject-id in a cookie that will identify the user to the validation web service with the web browser. Hence, the second attempt to visit `http://opentox.informatik.uni-freiburg.de/validation/report/validation/14` is successful. The user is provided with the validation report (Figure 30).

⁶¹ Login screen for validation service: <http://opentox.informatik.uni-freiburg.de/validation/login>

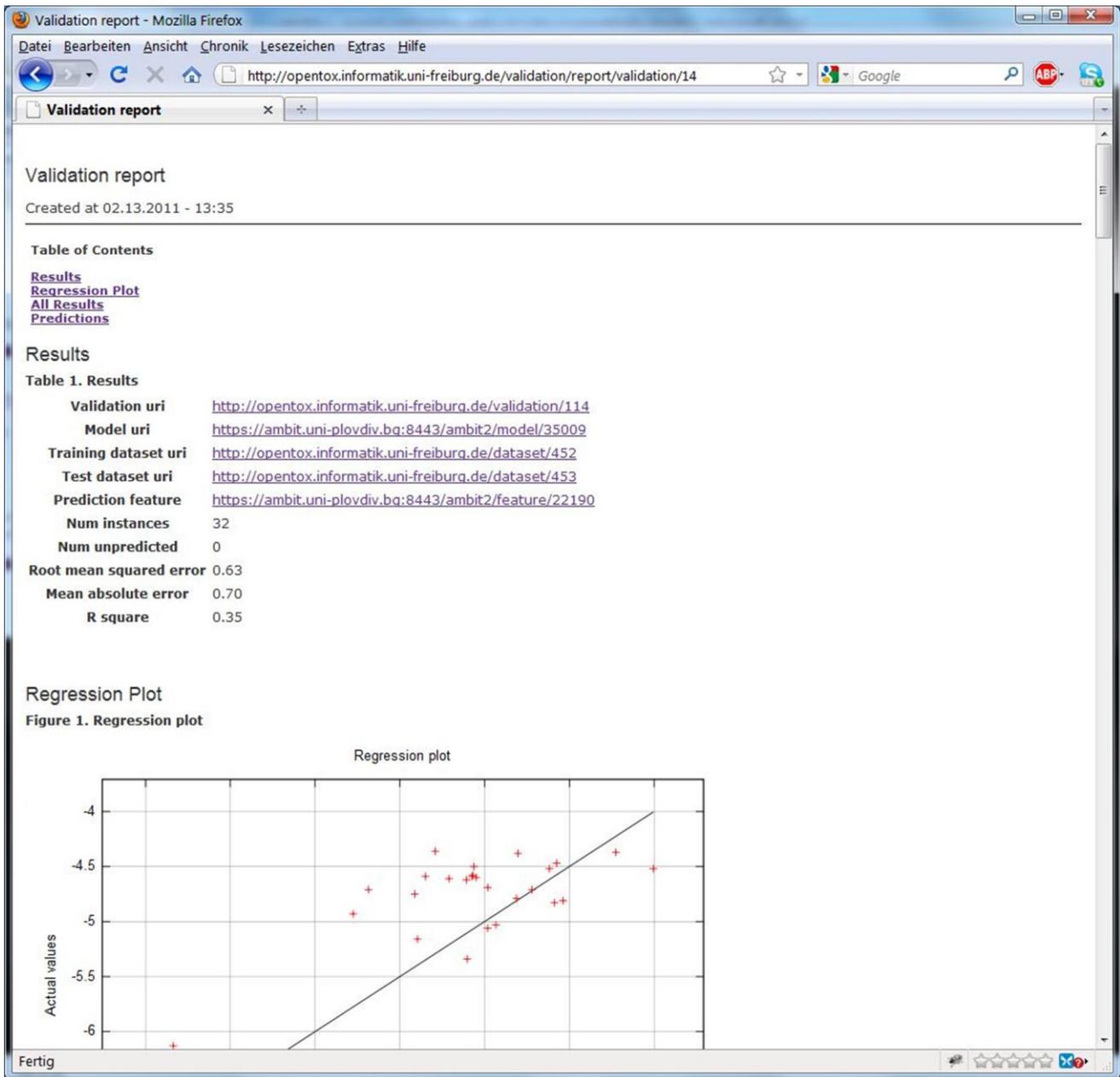


Figure30: Successful access to the validation report

7 Validation against confidential data using standalone version

7.1 Standalone Installation of OpenTox Web Services

A distributed system over the Internet has convenience and extensibility advantages due to its high flexibility and easy integration of new services. However, its security is questioned by certain users, because even when encrypted, data are still transferred over the Internet and it might be possible for someone to eavesdrop the communication and steal sensitive information (Figure 31). The authorization and authentication strategy adopted in OpenTox and the overall security system provide a high level of protection of confidential data. We understand, however, that toxicity data can be considered highly confidential and sensitive by their owners and even a slight possibility of leakage might be an obstacle for potential end users of OpenTox services and applications.

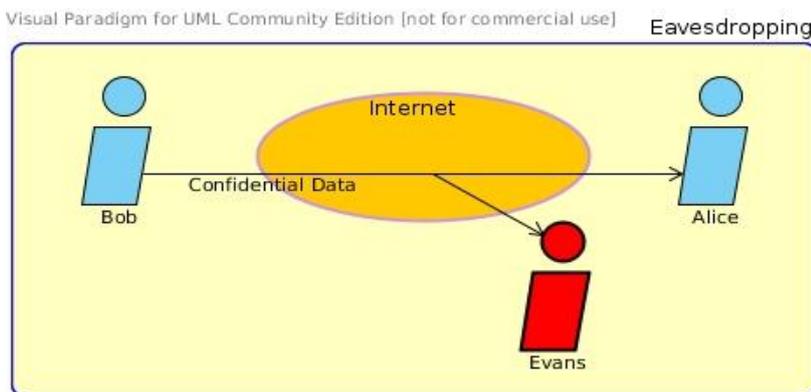


Figure 31: Eavesdropping of sensitive information

In order to minimize as much as possible the risk of data leakage, OpenTox offers an alternative implementation of the use case, which is based on a stand-alone local installation of some or all OpenTox services. The alternative approach is considered as the most secure way to seal data, namely to protect them physically prohibiting any kind of interaction with others and restraining their mobility within an isolated system. Complete physical isolation of the system means that the application runs on a machine either disconnected from the Internet (or any other network) or protected by means of firewalls. A virtual private network can also be established (Figure 32), again isolating the nodes of the distributed application from the rest of the network and also protecting the client-server communication using secure cryptographic tunnelling protocols.

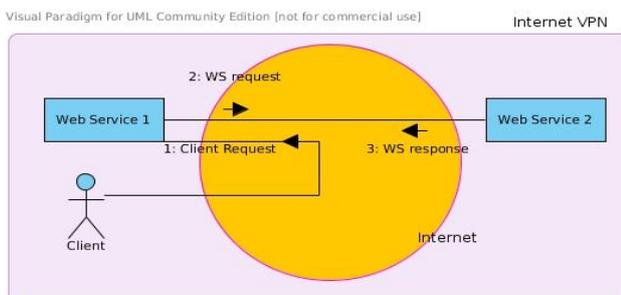


Figure 32: Topological structure of a virtual private network established over the Internet or some Local Area Network

The local installation of OpenTox web services along with web interfaces that facilitate their consumption is feasible since all projects under OpenTox are freely distributed (executables, documentation and source code) and are available on-line from the www.opentox.org website⁶². All service providers offer the ability to download and install individual web service implementations locally either as standalone applications or in a Servlet container (such as Apache Tomcat⁶³). In the case of a servlet container, the web service implementations come as “web archive” files (.war).

In all cases documentation is provided regarding the installation of prerequisites such as the **MySQL database server** or a **J2EE-compatible servlet container** such as Apache Tomcat.

7.2 Standalone Installation of particular services

This section describes how a user can install locally three OpenTox services namely AMBIT, Jaqpot, and ToxCreate. The AMBIT web service package is one of the several existing independent implementations of the OpenTox Application Programming Interface and is built according to the principles of the Representational State Transfer (REST) architecture. The Open Source Predictive Toxicology Framework, developed by partners of the EC FP7 OpenTox project, aims at providing a unified access to toxicity data and predictive models, as well as validation procedures. This is achieved by i) an information model, based on a common OWL–DL ontology; ii) links to related ontologies; iii) data and algorithms, available through a standardized REST web services interface, where every compound, data set or predictive method has a unique web address, used to retrieve its Resource Description Framework (RDF) representation, or initiate the associated calculations.

The Jaqpot web services are OpenTox API 1.2-compliant web services. Jaqpot is a web application that supports model training and data preprocessing algorithms such as multiple linear regression, support vector machines, neural networks (an in-house implementation based on an efficient algorithm), an implementation of the leverage algorithm for domain of applicability estimation and various data preprocessing algorithms such as PLS and data cleanup. Jaqpot also comes with a web service for storing BibTeX⁶⁴ entries which become also available in JSON and RDF formats. Jaqpot provides asynchronous execution of tasks submitted by users, authentication, authorization and accounting mechanisms powered by OpenSSO and two monitoring access points mounted at /monitoring and /status.

ToxCreate is a QSAR web application that has been developed in OpenTox. It derives nearest neighbours of the query structure and uses those to learn a model. Currently, it is being extended to accommodate any OpenTox-compliant model and dataset service.

7.2.1 AMBIT

The user downloads the AMBIT 2.0 application (<http://www.ideaconsult.net/downloads/ambit2/ambit2.war>) and saves the file `ambit2.war`. With his web browser, he navigates to <http://localhost:8080>, and clicks on “Tomcat Manager” in the Administration box at the top-left of the screen. He is prompted to enter the user name and password of the Tomcat manager/administrator he has set up. On the manager page, he scrolls to the bottom and finds the box entitled “WAR file to deploy”⁶⁵.

⁶² OpenTox downloads: <http://www.opentox.org/downloads>

⁶³ Apache Tomcat home page: <http://tomcat.apache.org/index.html>

⁶⁴ BibTeX specifications online: <http://www.bibtex.org/>

⁶⁵ More documentation regarding deployment on a Tomcat servlet container can be found online at <http://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html>

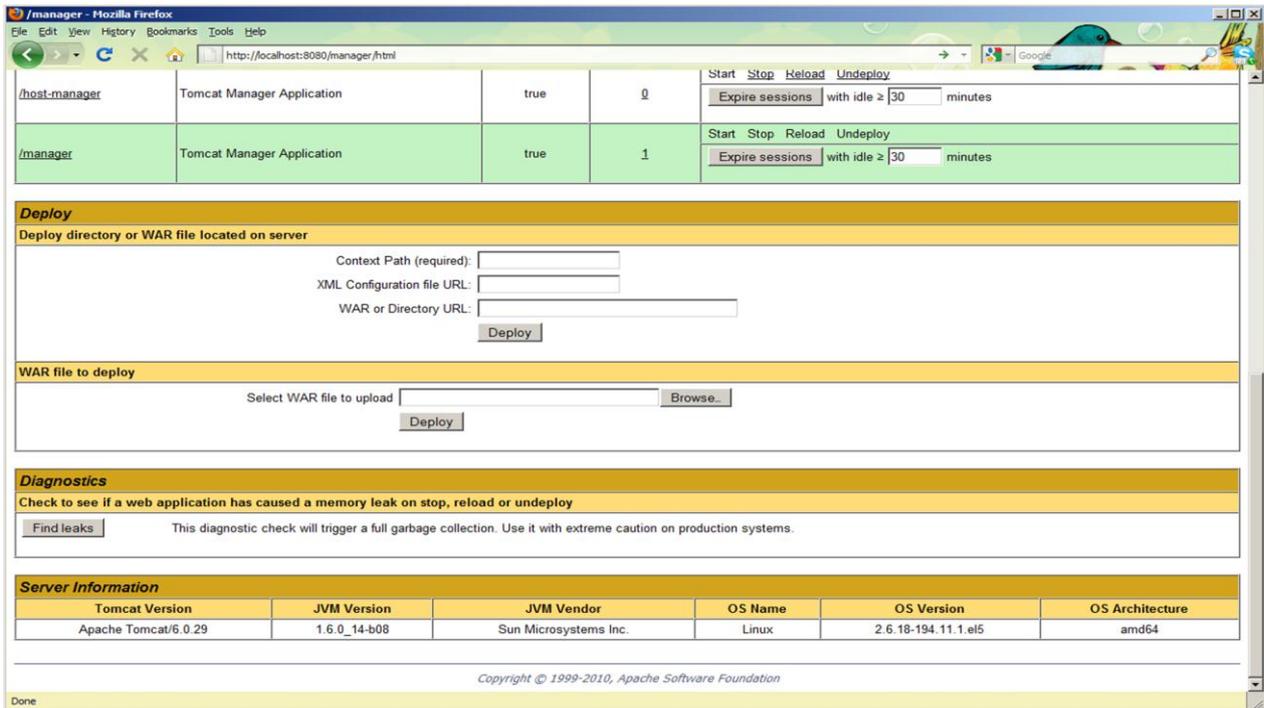


Figure 33: Screenshot from the deployment of AMBIT on a tomcat servlet container

Under “WAR file to deploy”, he clicks “Browse...”, finds ambit2.war and clicks “Deploy”. Following these steps, he has successfully installed the AMBIT 2.0 implementation of the OpenTox REST API. If he next navigates to <http://localhost:8080/abmit2> he should see the welcome screen of AMBIT2. As explained in the installation instructions for AMBIT 2.0 (ambit.sourceforge.net), this release (September 2010) comes without an embedded database (Figure 33).

An empty database can be created using cURL (curl.haxx.se). On many linux systems, cURL can be easily installed from a package repository using a standard package manager. It can also be downloaded from <http://curl.haxx.se/download.html>. Under Windows, there are two options for using cURL: 1) installing cURL natively, preferably using the most recent generic Win32 version: <http://curl.haxx.se/download.html>, or 2) installing the VMWare Player (see online <http://www.vmware.com/products/player/>) and running a small Linux environment (<http://www.maunz.de/opentox/dsl-4.1.zip>) under Windows (after installing VMWare Player and unpacking the dsl-4.1.zip file, just double-click the dsl-4.1.vmx file). Under Linux, after installing cURL, the following command can be typed as root in a console:

```
$ curl -X POST -d "dbname=ambit2" -d "user=mysqladminuser" -d "pass=mysqladminpass" http://localhost:8080/ambit2/admin/database
```

7.2.2 Jaqpot

A user can download and install Jaqpot on their local machine following these instructions (Tested on Ubuntu, Debian and Mac OS X operating systems). First, he needs to install on his system the following:

1. MySQL database server and client
2. Maven2
3. Git

The dependencies #2 and #3 are optional but will facilitate a lot the installation of Jaqpot. The commands needed for downloading the latest version of Jaqpot are the following:

```
$git clone git://github.com/alphaville/jaqpot.git
$cd jaqpot/
$mvn clean package tomcat:run
```

and Jaqpot will start on port 8080 (see <http://localhost:8080/jaqpot>).

7.2.3 ToxCreate

ToxCreate is distributed as a ready-to-use virtual machine (appliance). This offers several advantages over traditional installers:

- The appliance is deployed anywhere in minutes, not even administrative privileges are needed on a Windows machine⁶⁶.
- The appliance can be deployed, as is, on virtual servers or cloud-based services. By extracting the filesystem from the virtual hard disk, the Linux operating system is also installed quickly on a physical machine.

A local installation of ToxCreate is a fully functional QSAR solution and offers complete privacy. Installation instructions and downloads can be found on: <https://github.com/helma/opentox-documentation/wiki/Installation-of-IST-OpenTox-webservices>

8 Conclusions

This report summarizes the work that has been accomplished within the OpenTox Framework on the definition and implementation of web services supporting rapid prototyping for the generation of REACH relevant documents for validation in the form of standardized reports for (Q)SAR-based predictive toxicology models and their predictions. The validation and reporting framework was implemented as RESTful web services, and is available as open source applications in public repositories. The REST Web service architecture allows sharing of data and functionality among loosely-coupled, heterogeneous systems, such that a seamless workflow is possible using different system setups.

The validation services evaluate the performance of prediction algorithms. This is done by building models with training datasets, and applying those prediction models to test datasets. The predicted values are compared to the actual known outcome. The validation services in OpenTox provide a detailed description of the result of such a validation. Furthermore, the enhanced QMRF Editor and the novel QPRF editor (Q-edit) support the submission of the validation results to legislators.

The framework also supports validation against confidential data employing two distinct approaches: namely, using authentication and authorisation (A&A) and by providing a standalone version of the complete OpenTox services. Within this document we have described in detail, how A&A is implemented and used within the framework. And finally we have shown two use-cases for the validation of confidential data.

⁶⁶ See VirtualBox portable version, <http://www.vbox.me>

9 Appendix

9.1 Training test split report

Validation report

Created at 21.07.2011 – 11:30

Table of Contents

Results
 Confusion Matrix
 Plots
 All Results
 Predictions

Results

Table 1. Results

Validation uri	http://opentox.informatik.uni-freiburg.de/validation/451
Model uri	http://opentox.informatik.uni-freiburg.de/model/175
Training dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/1450
Test dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/1451
Prediction feature	http://opentox.informatik.uni-freiburg.de/dataset/333/feature/SAL
Num instances	273
Num unpredicted	22
Accuracy	0.733
Weighted accuracy	0.807
Weighted area under roc	0.634
Area under roc	true: 0.695, false: 0.583
F measure	true: 0.712, false: 0.751
True positive rate	0.728
True negative rate	0.737

Confusion Matrix

Table 2. Confusion Matrix

		actual		
		true	false	total
predicted	true	83	36	119
	false	31	101	132
total		114	137	

Plots

Table 3. Plots for all predictions

Figure 1. ROC Plot

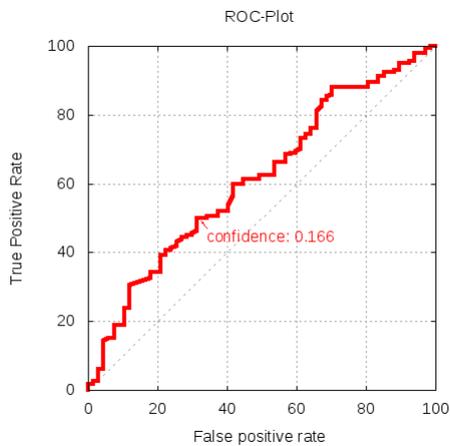


Figure 2. Percent Correct vs Confidence Plot

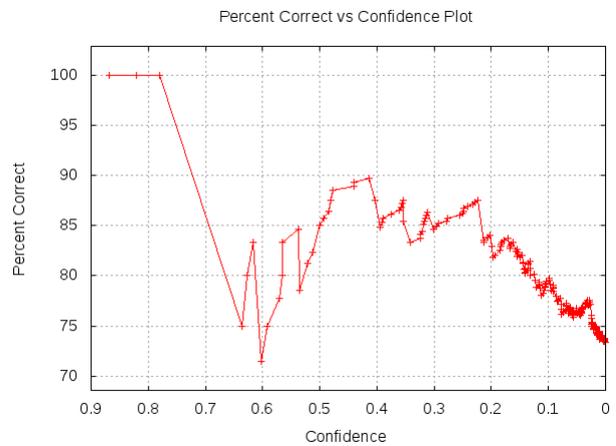


Table 4. Plots for predicted class-value 'true'

Figure 3. ROC Plot

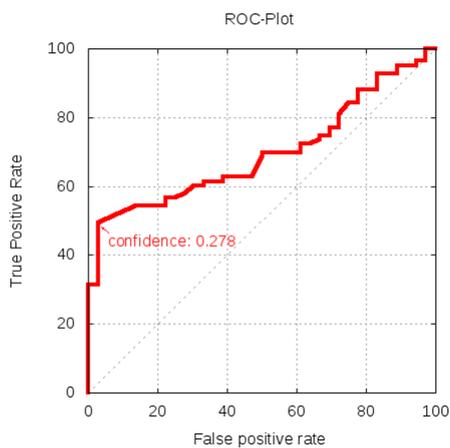


Figure 4. Percent Correct vs Confidence Plot

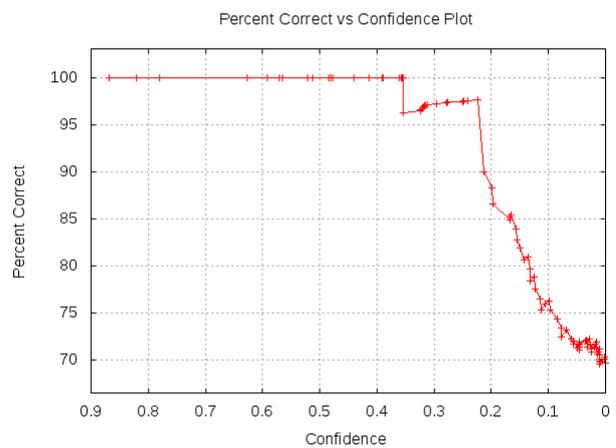
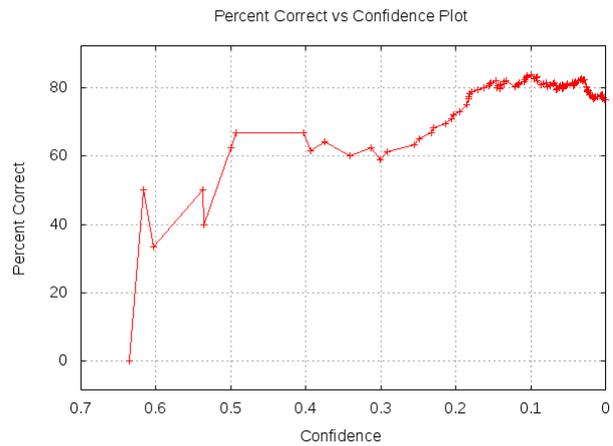
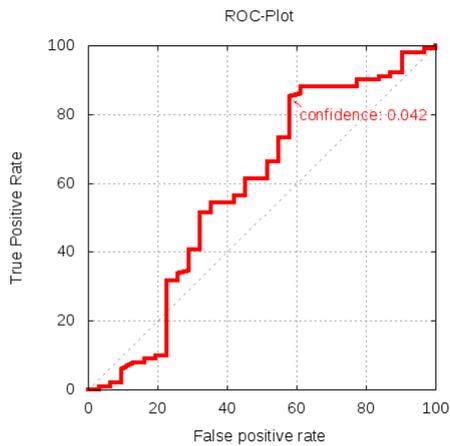


Table 5. Plots for predicted class-value 'false'

Figure 5. ROC Plot



All Results

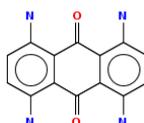
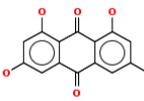
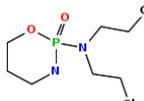
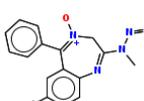
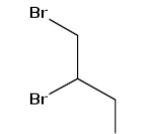
Table 6. All Results

Validation uri	http://opentox.informatik.uni-freiburg.de/validation/451
Validation type	training_test_split
Model uri	http://opentox.informatik.uni-freiburg.de/model/175
Algorithm uri	http://opentox.informatik.uni-freiburg.de/algorithm/lazar
Training dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/1450
Prediction feature	http://opentox.informatik.uni-freiburg.de/dataset/333/feature/SAL
Test dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/1451
Test target dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/333
Prediction dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/1453
Date	Thu Jul 21 11:25:55 +0200 2011
Num instances	273
Num without class	0
Num unpredicted	22
Real runtime	170.141883134842
Percent without class	0.0
Percent unpredicted	8.05860805860806
Num correct	184
Num incorrect	67
Confusion matrix	confusion_matrix_predicted: true, confusion_matrix_actual: true: 83, confusion_matrix_predicted: false, confusion_matrix_actual: true: 31, confusion_matrix_predicted: true, confusion_matrix_actual: false: 36, confusion_matrix_predicted: false, confusion_matrix_actual: false: 101
Percent correct	73.307
Percent incorrect	26.693

Weighted area under roc	0.634
Accuracy	0.733
Weighted accuracy	0.807
Num false positives	36
Num false negatives	31
Num true positives	83
Num true negatives	101
Area under roc	true: 0.695, false: 0.583
False negative rate	0.272
False positive rate	0.263
F measure	true: 0.712, false: 0.751
Precision	true: 0.697, false: 0.765
True negative rate	0.737
True positive rate	0.728

Predictions

Table 7. Predictions

compound	actual value	predicted value	classification	confidence value	compound-uri
	true	true		0.869	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C14H12N4O2/c15-5-1-2-6(16)10-9(5)13(19)11-7(17)3-4-8(18)12(11)14(10)20/h1-4H,15-18H2
	true	true		0.821	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C15H10O5/c1-6-2-8-12(10(17)3-6)15(20)13-9(14(8)19)4-7(16)5-11(13)18/h2-5,16-18H,1H3
	true	true		0.780	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C7H15Cl2N2O2P/c8-2-5-11(6-3-9)14(12)10-4-1-7-13-14/h1-7H2,(H,10,12)
	true	false		0.635	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C16H14ClN4O2/c1-20(19-22)15-10-21(23)16(11-5-3-2-4-6-11)13-9-12(17)7-8-14(13)18-15/h2-9,23H,10H2,1H3/q+1
	true	true		0.628	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C3H6Br2O/c4-1-3(5)2-6/h3,6H,1-2H2

9.2 Cross-validation report

Crossvalidation report

Created at 21.06.2011 – 13:33

Table of Contents

- Crossvalidation Results
- Plots
- Results
- All Results
- Predictions

Crossvalidation Results

These performance statistics have been derived by accumulating all predictions on the various fold (i.e. these numbers are NOT averaged results over all crossvalidation folds).

Table 1. Crossvalidation Results

Crossvalidation uri	http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/18
Algorithm uri	http://opentox.informatik.uni-freiburg.de/algorithm/lazar
Dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/556
Num folds	10
Num instances	569
Num unpredicted	10
Root mean squared error	37.97
Mean absolute error	6.46
R square	0.20

Plots

Figure 1. Regression plot

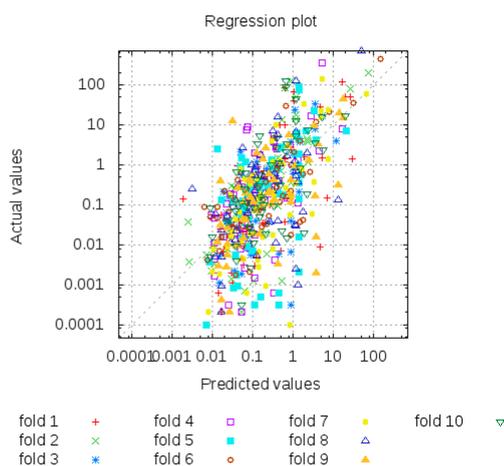


Figure 2. Percent Correct vs Confidence Plot

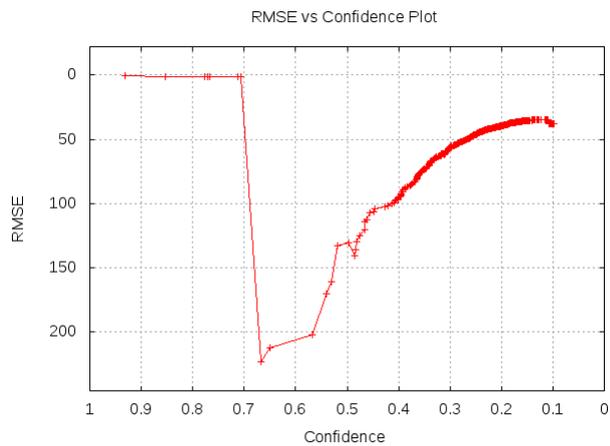
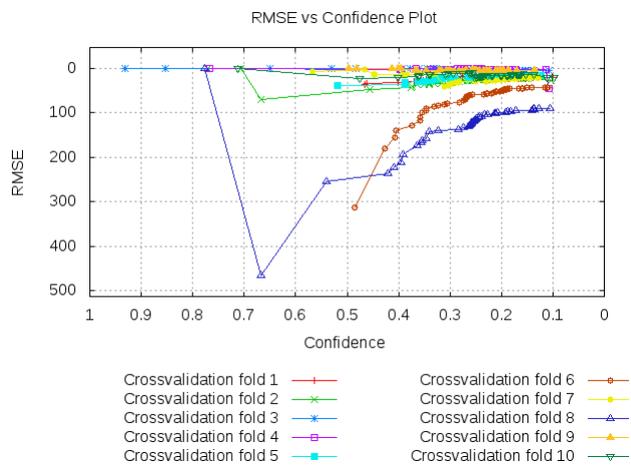


Figure 3. Percent Correct vs Confidence Plot



Results

Table 2. Results

Validation uri	Crossvalidation fold	Num instances	Num unpredicted	Root mean squared error	Mean absolute error	R square
http://opentox.informatik.uni-freiburg.de/validation/172	1	57	0	21.09	7.64	0.10
http://opentox.informatik.uni-freiburg.de/validation/173	2	57	1	28.06	9.10	0.32
http://opentox.informatik.uni-freiburg.de/validation/174	3	57	0	5.39	1.71	0.12
http://opentox.informatik.uni-freiburg.de/validation/175	4	57	0	46.45	7.38	4.17e-03
http://opentox.informatik.uni-freiburg.de/validation/176	5	57	0	15.35	4.06	-0.040

Validation uri	Crossvalidation fold	Num instances	Num unpredicted	Root mean squared error	Mean absolute error	R square
http://opentox.informatik.uni-freiburg.de/validation/177	6	57	2	42.50	7.05	0.52
http://opentox.informatik.uni-freiburg.de/validation/178	7	57	3	19.77	4.77	0.15
http://opentox.informatik.uni-freiburg.de/validation/180	8	57	1	89.69	15.14	0.11
http://opentox.informatik.uni-freiburg.de/validation/181	9	57	3	4.71	1.80	0.54
http://opentox.informatik.uni-freiburg.de/validation/182	10	56	0	21.61	5.84	-0.055

All Results

Table 3. All Results⁶⁷

Validation uri	http://opentox.informatik.uni-freiburg.de/validation/172	http://opentox.informatik.uni-freiburg.de/validation/173	http://opentox.informatik.uni-freiburg.de/validation/174	http://opentox.informatik.uni-freiburg.de/validation/175	http://opentox.informatik.uni-freiburg.de/validation/176
Validation type	crossvalidation	crossvalidation	crossvalidation	crossvalidation	crossvalidation
Model uri	http://opentox.informatik.uni-freiburg.de/model/133	http://opentox.informatik.uni-freiburg.de/model/134	http://opentox.informatik.uni-freiburg.de/model/135	http://opentox.informatik.uni-freiburg.de/model/136	http://opentox.informatik.uni-freiburg.de/model/137
Algorithm uri	http://opentox.informatik.uni-freiburg.de/algorithm/lazar				
Training dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/558	http://opentox.informatik.uni-freiburg.de/dataset/560	http://opentox.informatik.uni-freiburg.de/dataset/562	http://opentox.informatik.uni-freiburg.de/dataset/564	http://opentox.informatik.uni-freiburg.de/dataset/566
Prediction feature	http://opentox.informatik.uni-freiburg.de/dataset/556/feature/LC_50_mmol				
Test	http://opentox.informatik.uni-freiburg.de/validation/172	http://opentox.informatik.uni-freiburg.de/validation/173	http://opentox.informatik.uni-freiburg.de/validation/174	http://opentox.informatik.uni-freiburg.de/validation/175	http://opentox.informatik.uni-freiburg.de/validation/176

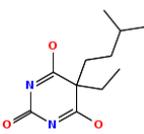
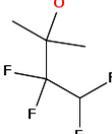
⁶⁷ Modified to fit the page: 5 fold columns removed

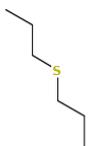
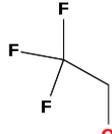
dataset uri	ormatik.uni- freiburg.de/dataset t/559	ormatik.uni- freiburg.de/dataset t/561	ormatik.uni- freiburg.de/dataset t/563	ormatik.uni- freiburg.de/dataset t/565	ormatik.uni- freiburg.de/dataset t/567
Test target	http://opentox.inf	http://opentox.inf	http://opentox.inf	http://opentox.inf	http://opentox.inf
dataset uri	ormatik.uni- freiburg.de/dataset t/556	ormatik.uni- freiburg.de/dataset t/556	ormatik.uni- freiburg.de/dataset t/556	ormatik.uni- freiburg.de/dataset t/556	ormatik.uni- freiburg.de/dataset t/556
Prediction	http://opentox.inf	http://opentox.inf	http://opentox.inf	http://opentox.inf	http://opentox.inf
dataset uri	ormatik.uni- freiburg.de/dataset t/579	ormatik.uni- freiburg.de/dataset t/581	ormatik.uni- freiburg.de/dataset t/583	ormatik.uni- freiburg.de/dataset t/585	ormatik.uni- freiburg.de/dataset t/587
Date	Tue Jun 21 13:06:03 +0200 2011	Tue Jun 21 13:08:47 +0200 2011	Tue Jun 21 13:11:26 +0200 2011	Tue Jun 21 13:13:59 +0200 2011	Tue Jun 21 13:16:33 +0200 2011
Num instances	57	57	57	57	57
Num without class	0	0	0	0	0
Num unpredicted	0	1	0	0	0
Real runtime	151.7974388599 4	146.4431538581 85	140.8381531238 56	140.7759420871 73	158.5764129161 83
Percent without class	0.0	0.0	0.0	0.0	0.0
Percent unpredicted	0.0	1.754385964912 28	0.0	0.0	0.0
Crossvalidation	18	18	18	18	18
Crossvalidation uri	http://opentox.inf ormatik.uni- freiburg.de/valida tion/crossvalidatio n/18	http://opentox.inf ormatik.uni- freiburg.de/valida tion/crossvalidatio n/18	http://opentox.inf ormatik.uni- freiburg.de/valida tion/crossvalidatio n/18	http://opentox.inf ormatik.uni- freiburg.de/valida tion/crossvalidatio n/18	http://opentox.inf ormatik.uni- freiburg.de/valida tion/crossvalidatio n/18
Crossvalidation fold	1	2	3	4	5
Root mean squared error	21.09	28.06	5.39	46.45	15.35
Mean absolute	7.64	9.10	1.71	7.38	4.06

error					
R square	0.10	0.32	0.12	4.17e-03	-0.040
Target	505.54	1182.66	33.69	2204.88	230.57
variance					
actual					
Target	38.55	109.22	3.04	5.48	8.11
variance					
predicted					
Sum	25347.72	44080.13	1656.00	122958.13	13433.26
squared					
error					
Sample	0.46	0.80	0.43	0.28	0.11
correlation					
coefficient					
Dataset uri	http://opentox.informatik.uni-freiburg.de/dataset/556				
Num folds	10	10	10	10	10
Stratified	false	false	false	false	false
Random seed	1	1	1	1	1

Predictions

Table 4. Predictions

compound	actual value	predicted value	confidence value	compound-uri
	0.38	0.14	0.93	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C11H18N2O3/c1-4-11(6-5-7(2)3)8(14)12-10(16)13-9(11)15/h7H,4-6H2,1-3H3,(H2,12,13,14,15,16)
	3.63	1.45	0.85	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C5H8F4O/c1-4(2,10)5(8,9)3(6)7/h3,10H,1-2H3
	0.20	0.19	0.78	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C11H18N2O3.Na/c1-4-6-7(3)11(5-2)8(14)12-10(16)13-9(11)15;/h7H,4-6H2,1-3H3,(H2,12,13,14,15,16);/q;+1/p-1
	0.034	0.053	0.77	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C8H18S2/c1-3-9-7-5-6-8-10-4-2/h3-8H2,1-2H3

compound	actual value	predicted value	confidence value	compound-uri
	0.18	0.027	0.77	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C6H14S/c1-3-5-7-6-4-2/h3-6H2,1-2H3
	0.091	0.34	0.71	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C12H18N2O3.Na/c1-4-6-8(3)12(7-5-2)9(15)13-11(17)14-10(12)16;/h5,8H,2,4,6-7H2,1,3H3,(H2,13,14,15,16,17);/q:+1/p-1
	1.19	2.30	0.71	http://opentox.informatik.uni-freiburg.de/compound/InChI=1S/C2H3F3O/c3-2(4,5)1-6/h6H,1H2

9.3 Algorithm comparison report

Algorithm comparison report

Created at 21.07.2011 – 10:57

Table of Contents

Dataset: <http://apps.ideaconsult.net:8080/ambit2/dataset/603306>

Average Results on Folds

Bar Plot

Paired t-test

Dataset: <http://apps.ideaconsult.net:8080/ambit2/dataset/603306>

Average Results on Folds

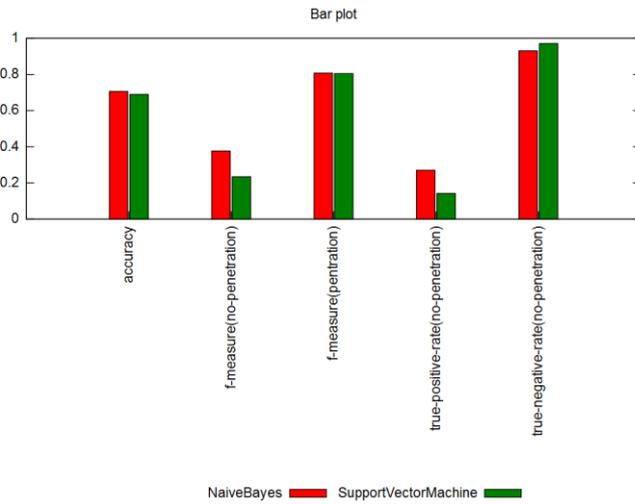
These performance statistics have been derived by computing the mean of the statistics on each crossvalidation fold.

Table 1. Average Results on Folds

Identifier	NaiveBayes	SupportVectorMachine
Crossvalidation uri	http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/47	http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/49
Crossvalidation report uri	http://opentox.informatik.uni-freiburg.de/validation/report/crossvalidation/39	http://opentox.informatik.uni-freiburg.de/validation/report/crossvalidation/49
Algorithm uri	http://apps.ideaconsult.net:8080/ambit2/algorithm/NaiveBayes	http://apps.ideaconsult.net:8080/ambit2/algorithm/SMO
Num instances	83.000	83.000
Num unpredicted	0	0
Accuracy	0.706 ± 2.58e-03	0.689 ± 1.02e-04
Weighted accuracy	0 ± 0	0 ± 0
Weighted area under roc	0 ± 0	0 ± 0
Area under roc	penetration: 0 ± 0, no-penetration: 0 ± 0	penetration: 0 ± 0, no-penetration: 0 ± 0
F measure	penetration: 0.807, no-penetration: 0.376	penetration: 0.805, no-penetration: 0.234
True positive rate	0.270	0.142
True negative rate	0.931	0.971

Bar Plot

Figure 1. Bar Plot



Paired t-test

Table 2. percent_correct, significance-level: 0.9, num results: 5

NaiveBayes SupportVectorMachine

NaiveBayes

SupportVectorMachine

Table 3. weighted_area_under_roc, significance-level: 0.9, num results: 5

NaiveBayes SupportVectorMachine

NaiveBayes

SupportVectorMachine