Deliverable D 4.2

Report on Initial Prototype of (Q)SAR Algorithms

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

Contract No.	Health-F5-2008-200787
Document Type:	Deliverable Report
WP/Task:	WP4
Name	Report on Initial Prototype of (Q)SAR Algorithms
Document ID:	OpenTox Deliverable Report WP4
Date:	March 8, 2010
Status:	Final Version
Organisation:	National Technical University of Athens (NTUA)

Contributors		
	Haralambos Sarimveis	NTUA
	Pantelis Sopasakis	NTUA
	Stefan Kramer	TUM
	Tobias Girschick	TUM
	Fabian Buchwald	TUM
	Nina Jeliazkova	IDEA
	Christoph Helma	IST
	David Vorgrimmler	IST
	Barry Hardy	DC
	Nicki Douglas	DC
	Alexey Zakharov	IBMC
	Sunil Chawla	SL-JNU
	Surajit Ray	SL-JNU

Distribution:	Public

Purpose of Document:	To document results for this deliverable

Document History:	1 – Initial draft prepared on Feb 2, 2010
	2 – Second draft prepared on Feb 16, 2010
	3 – Third draft prepared on Feb 21, 2010

	4 – Fourth draft prepared on Feb 22, 2010 5 – Fifth draft prepared on Feb 24, 2010
	6 – Sixth draft prepared on Feb 25, 2010
	7 – Seventh draft prepared on Feb 26, 2010
	8 – Final Version approved, Feb 28 2010
	9 – Updated Version, March 8 2010

Table of Contents

Summary

Quantitative structure-activity relationship (QSAR) algorithms are used in predictive toxicology to develop models that correlate structural, chemical and biological properties of chemical compounds with toxicological endpoints. QSAR algorithms have advanced in recent years from simple linear regression methods to sophisticated methodologies developed by researchers in the scientific areas of statistics, computer science (machine learning and computational intelligence), cheminformatics and bioinfomatics. The development of supporting algorithms (algorithms for calculating structural properties, selecting subsets of descriptors, estimating the applicability domain and measuring chemical similarities) and consensus modelling approaches have further improved the acceptance and reliability of QSAR analysis. Development of QSAR algorithms is an active research area and recent advances have extended applications into other scientific disciplines, including chemical biology, QSAR-directed virtual screening and systems biology.

The overall objective of the OpenTox Algorithms Work Package is to define and implement the integration of state-of-the-art QSAR and supporting algorithms into the OpenTox Framework. The Framework was designed aligned with the key OpenTox principles of interoperability, extensibility and compliance with user requirements and use cases. Special care was taken to allow independent development of different algorithm software components, which will enable the integration of additional new algorithms in the future that may be offered not only by OpenTox partners but also by many third-party developers in the community. The Framework works independently of the underlying data and thus, although emphasis is given to the prediction of the REACH relevant end-points (e.g., chronic and reproductive toxicity, *in vivo* mutagenicity and carcinogenicity), it can easily be extended to predict additional endpoints, such as ecotoxicity.

This report on the initial prototype of (Q)SAR Algorithms presents the work that has been accomplished within the OpenTox project on the implementation of QSAR and supporting algorithms during the first 18 months of the project. Prototype development has followed the RestFul Web Service architecture so that services comply with the current version of the OpenTox standardized interfaces and algorithm ontology. The services include cheminformatics, statistical and data mining tools that have been provided by OpenTox partners as well as popular and well-accepted algorithms contained in other state-of-the-art open source projects (e.g. WEKA, CDK). To be consistent with the open source philosophy of the OpenTox FP7 research project, open source tools have been utilized for developing the initial algorithm prototypes and all source code is available to the public. Due to the use of standardized interfaces, commercial software modules may additionally be incorporated in applications based on the platform as component web services.

This report provides a comprehensive and detailed documentation of all the algorithms that have already been implemented within OpenTox, thus serving as a valuable knowledge source for predictive toxicology application development and deployment.

1. Introduction

Ongoing scientific efforts in various complementary fields have led to a large number of algorithms that are available and potentially useful for (Q)SAR and related tasks. During the first six months of the OpenTox project, a comprehensive review of available algorithms was performed by project partners. The review considered algorithms that have been developed and can be provided by project participants. The algorithms were grouped into four categories: *descriptor calculation algorithms, classification and regression algorithms, feature selection algorithms and algorithms for the aggregation of results from multiple QSAR models*. The deliverable report D4.1 on Algorithm Selection and Evaluation, submitted on February 28, 2009, gave a detailed description of these algorithms and, based on a set of multiple selection criteria, made a prioritization of the algorithms which indicated in which stage of the OpenTox project each algorithm is planned to be integrated into the OpenTox Framework. Algorithms with priority A were planned to be implemented in the first (Q)SAR prototype. As it will be shown in the report, this task has been successfully accomplished (Open Babel is the only set of priority A descriptor calculation algorithms that has not been prototyped yet as an individual software component, but it is implicitly used in the fminer descriptor calculation implementation for substructure matching). Additionally, a significant number of algorithms with lower priorities have also been prototyped.

The OpenTox project involves many partners and developers having different programming backgrounds and experience. A common collaboration framework based on the RestFul Web Service architecture was defined for algorithm implementation that supports independent deployment of services by different partners, which may be combined in an interoperable manner into OpenTox-based applications. This framework makes it convenient for third-party (Q)SAR and machine learning researchers and developers to integrate their own tools within the OpenTox Framework through well-established HTTP methods. The implementations are arranged in a way that requests are forwarded from one server to the other, minimizing latency time and boosting performance. Additionally, all communications are stateless in the sense that none of the nodes needs to store any data; requests are self-contained and provide all the data needed by the application to process them.

Algorithm prototypes are presented in a uniform tabular format, which gives brief algorithm descriptions, defines responsible partners and contact persons and provides links to the resources and the OpenTox Application Programming Interface (API). More technical information (such as input parameters and output results, status codes, programming languages, external libraries) is also included in each algorithm prototype description. Algorithm prototype descriptions are completed with examples that can be used to test and evaluate efficiency of the implementations.

2. Implementation principles

Algorithm prototypes constitute a major component of the overall OpenTox prototype and should be in line with the key OpenTox design principles of interoperability, extensibility and compliance with user requirements and use cases. Special care was taken to allow independent development of different algorithm software components. The OpenTox algorithm implementation principles are presented next:

2.1 Open Source programming tools

In order to be consistent with the open source philosophy aspect of the project, initial algorithm implementation was realized using open source programming tools. Java[1] was the main programming language used by most partners, but other languages such as Ruby[2] were utilized as well. The implementation process also involved the utilization of widely-used and well-accepted open source machine learning and data mining libraries, such as Weka[3]. This collection of tools permits full exploitation of opportunities coming from open source software. Future developments of the OpenTox Framework as an industry platform will also address support of the combination of non-open source and open source algorithm and modelling components and services into OpenTox-based applications, compliant with OpenTox APIs, interoperability standards and guidance.

2.2 Restful Web Service architecture

A Web service architecture was chosen for developing the algorithm prototype. Web services are rapidly emerging as a popular standard for sharing data and functionality among loosely-coupled, heterogeneous systems. In particular the Representational State Transfer (REST) Web service architecture[4] was chosen because of the following advantages:

1. The produced web services are stateless;
2. The produced web services have a uniform interface (the only allowed operations are the HTTP operations);
3. The resources are uniquely identified by URIs and described by representations;
4. Components manipulate resources by exchanging representations of the resources.

All algorithm resources have representations providing information about the type of algorithm, what the algorithm accepts as input, the tuning parameters etc. Most algorithms and model resources in OpenTox are available in multiple representations. The Resource Description Framework (RDF) representation[5], and in particular its XML-formatted variant, was chosen as the master data exchange format, due to the following reasons:

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 humans can, in principle, read RDF documents, it is unlikely that they are able to understand them easily. RDF is intended to be understood by computers, not people.

[1] java.net/
[2] www.ruby-lang.org/en/
[3] www.cs.waikato.ac.nz/ml/weka/
[4] www.ibm.com/developerworks/webservices/library/ws-restful/
[5] www.w3.org/RDF/

Some services support additional representations like JavaScript Object Notation (JSON)[6], YAML[7] or Application/X-Turtle[8] . Some prediction model services provide Predictive Model Markup Language (PMML) representations (designed by DMG[9]) to improve their portability, since many machine learning applications like KNIME[10] and Weka provide support for PMML.

Examples of the structure of an Algorithm RDF document can be found at the OpenTox address: opentox.org/data/documents/development/RDF%20files/Algorithm .

2.3 OpenTox algorithm ontology

A human-readable format of the algorithm ontology, including a graphical representation can be found at opentox.org/dev/apis/api-1.1/Algorithms. A graphical overview of the current AlgorithmType ontology is included in the Appendix in this report.

A formal OWL[11] representation of the algorithm ontology is available at opentox.org/data/documents/development/RDF files/AlgorithmTypes/view.

The plan is to extend this ontology in the future to a full description of every algorithm, including references, parameters and default values. This will be achieved by adopting the Blue Obelisk ontology[12] and is currently work-in-progress.

The RDF representation of an Algorithm contains metadata described by the Dublin Core Specifications[13] for modeling metadata (DC Namespace) and the OpenTox namespace.

The establishment of an ontological base for the services facilitates the extension of the services and the introduction of new algorithms and new algorithm classes.

2.4 OpenTox algorithm and model Application Programming Interfaces (APIs)

Algorithm and Model APIs are part of the OpenTox API that enables interaction among all OpenTox software components. The current OpenTox API version is API 1.1 (www.opentox.org/dev/apis/api-1.1). Based on the REST and RDF principles mentioned before, each algorithm and each model, in RESTful terms, is a resource. Specifically, a representation of an algorithm contains information about the input a client should provide (obligatorily or optionally) to invoke the underlying procedure (e.g. training, data preprocessing etc).

Hereinafter the notation /resource will be used to denote the class of URIs someDomain.com/resource where someDomain.com can be the domain name of any OpenTox server (such as opentox.ntua.gr:3000,

---

[6] www.json.org/

[7] www.yaml.org/

[8] www.w3.org/TeamSubmission/turtle/

[9] www.dmg.org

[10] www.knime.com

[11] www.w3.org/TR/owl-features/

[12] qsar.svn.sf.net/viewvc/qsar/trunk/qsar-dicts/descriptor-ontology.owl?revision=218

[13] dublincore.org/

opentox.informatik.tu-muenchen.de:8080 and ambit.uni-plovdiv.bg:8080). All algorithm resources are placed under /algorithm and all model resources under /model. For example opentox.ntua.gr:3000/algorithm/svm is the resource of a Support Vector Machine algorithm and opentox.ntua.gr:3000/model/4538 is a model resource.

The **algorithm API** consists of a number of operations that are described next. Each operation uses one HTTP method (GET, PUT, POST or DELETE)[14]:

1. GET /algorithm: Returns a list of all available algorithms on the server in a supported media type; these include text/uri-list and rdf-related media types like application/rdf+xml, application/x-turtle, text/x-triple and text/rdf+n3. Optionally, a query can be added in the URI in the form '/algorithm?sameas={uri of other OpenTox algorithm}' or '/algorithm?type={uri of algorithm type specified by the OT algorithm type ontology}' to retrieve the algorithms that meet certain requirements.
2. GET /algorithm/{id}: Returns a representation of the algorithm, identified by its id, in a supported media type specified in the 'Accept' header of the request
3. POST /algorithm/{id}: A POST operation on an algorithm activates the application of the algorithm and often requires the specification of input parameters. For instance all prediction algorithms (machine learning or otherwise) need the parameter 'dataset_uri' which is the URI of the training dataset. Additionally the parameter 'prediction_feature' is mandatory for all supervised learning algorithms; it is the target feature of the provided dataset. The result from a successful POST operation is the URI of a created model. Supported media types for the result are all RDF-related and text/uri-list.

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

The same architectural concept was applied to the construction of the **model APIs**, which provide access to all OpenTox models.

- GET /model: Retrieve a complete list of models on the server. Additional queries may optionally be provided in the URI to get a certain subset of this collection. Examples include ?max=N to restrict the size of the list, ?sameas={uri of other model} to get a list of the models which are same as a given model. Other queries may be specified by the service providers.
- GET /model/{id}: Get the representation of a certain model in a supported media type. This representation contains information about the training algorithm that produced the model, the training dataset, the independent, dependent and predicted features of the model and the various training parameters (tuning parameters). An RDF representation is always available while a PMML format is provided for some classes of models and can be requested as 'application/xml'. Some services support additional media types such as JSON and YAML.
- GET /model/{id}/independent: Get the list of independent features of the model. Features are resources themselves so they are characterized by a URI of the form http://someserver.com/feature/{id}. The independent features of the model are the features of the training dataset, excluding the target. This list is available in all RDF-related media type (application/rdf+xml is mandatory) and in text/uri-list format as well.
- GET /model/{id}/dependent: Get the dependent feature of the model, that is the target feature of the training dataset specified in the "POST /algorithm/{id}" invocation by the 'prediction_feature' POSTed parameter.

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

- GET /model/{id}/predicted: This is a feature that is generated along with the creation of the model; it is the feature related to the predicted values of the dependent feature using this model.
- DELETE /model/{id}: A model can be deleted by users having proper authorization.
- POST /model/{id}: A dataset or a compound URI can be posted to a prediction model to get toxicological predictions. The service exploits the underlying model to calculate the predicted values and returns to the client (within the response body), a URI for the created dataset. This dataset contains the submitted compounds and their corresponding predictions.

More information about the model API is available at the address opentox.org/dev/apis/api-1.1/Model.

3. Progress achieved to date and future challenges

This section presents the progress achieved so far and the remaining challenges in algorithm prototyping in terms of the key OpenTox project goals:

3.1 Independent development of components

The initial OpenTox algorithm prototype consists of a set of software components which have already implemented more than the initially planned QSAR modelling and supporting algorithms. Six OpenTox partners (IDEA, IST, TUM, NTUA, IBMC, SL-JNU) have been involved in algorithm prototyping using different programming languages and having different programming backgrounds and experiences. The software components have been developed as functional building-blocks accessible over standard Internet protocols independent of platforms and programming languages.

3.2 Interoperability

Interoperability with respect to the OpenTox Framework refers to the principle that different OpenTox components or services may correctly exchange information with each other and subsequently make use of that information. Both syntactic interoperability for correct data exchange and semantic interoperability supporting the accurate communication of meaning and interpretation of data are supported principles for OpenTox resources. The establishment of a common API standard for all working groups and the introduction of the OpenTox ontology service have been important steps towards the desired interoperability between different services, which has been achieved to some extent (several examples that are presented in the detailed descriptions of algorithms later in this report integrate datasets and algorithm components from different servers to produce models). However, there are still many challenges to be addressed with regards to integrating algorithms into applications:

- Investigate further the content of the exchanges messages from the semantic point of view;
- Reduce the amount of data traveling over the web to increase the overall performance;
- Provide correct status codes and additional header information to allow the machines to communicate better with each other;
- Extend the API to include authorization/authentication mechanisms and to design a security level in a user friendly-way;
- A task runs on a server and lets the client be aware of the creation, progress and completion of a job. This way a server gains full control of the asynchronous jobs it runs, while it releases a socket to the client. Tasks are not yet fully implemented and this fact has slowed down the integration of services so far.

3.3 Extensibility

OpenTox needs to be extensible to a broad range of future algorithms and new functionalities that meet the current trends in QSAR, bioinformatics and predictive toxicology in general. The algorithm APIs allow developers to add new functionalities or to improve the existing ones without the need to amend the agreed standard. The REST architecture and the adoption of the RDF, as the main format for modeling meta-information (two of the most powerful state-of-the-art software design considerations) render the API truly extensible as most functionalities can be easily appended. An open issue that still remains a challenge regarding algorithm extensibility is the development and inclusion of consensus modeling techniques.

3.4 User requirements and use cases

The detailed description of a use case is fundamental to understand how the end user will interact with the web application and provide both a user-friendly interface and a robust and high performance underlying engine to meet user requirements. Two initial use cases have been identified for the implementation of the OpenTox Framework prototype. The first case, **ToxPredict**, is aimed at the user having no or little experience in QSAR predictions. This use case should offer an easy-to-use user interface, allowing the user to enter a chemical structure and to obtain in return a toxicity prediction for one or more endpoints. The second case, **ToxCreate**, is aimed at the experienced users, allowing them to construct and to validate models using a number of datasets and algorithms. Both use cases also demonstrate inter-connectivity between multiple partner services. Within **ToxPredict**, models produced from partners TUM, IDEA, and NTUA algorithm web services have been integrated, while in **ToxCreate** the model construction is performed using partner IST's algorithm web services. Important subsequent steps to be pursued in forthcoming development iterations are the integration of additional algorithm web services into the ToxCreate use case and the interoperation of ToxCreate and ToxPredict in terms of algorithm, modelling and validation services.

4. Algorithm prototype presentation

Algorithm web services are key components of the overall OpenTox framework and also important parts of the use cases, as they are responsible for data manipulation, descriptor calculation and selection, reduction of dimensionality, and most importantly generation of regression and classification (Q)SAR models. All algorithm services accept a dataset as input, so they assume the existence of a dataset service. Algorithms that have been included in the initial prototype are summarized next with reference to the categorization, selection and prioritization of algorithms, as these were presented in deliverable D4.1

The algorithms can be grouped in four categories: descriptor calculation algorithms, classification and regression algorithms, clustering algorithms and feature selection algorithms. Clustering algorithms form a new category of algorithms that was not considered in D4.1. A fifth category of algorithms (Algorithms for the aggregation of results from multiple QSAR models) are of Priority C and will be included only in the final prototype of (Q)SAR algorithms.

Descriptor calculation algorithms: This category currently includes algorithms which calculate descriptors that represent chemical structures. There are two different types of molecular descriptors, namely physico-chemical and (sub)structural descriptors. In the group of structural and sub-structural descriptors, five algorithms have been implemented (FreeTreeMiner, Fminer, gSpan, MakeMNA, MakeQNA). We have also prototyped two sets of descriptor calculation algorithms that belong to the group of physico-chemical descriptors, namely the Chemistry Development Toolkit (CDK) and JOELib. The initial (Q)SAR algorithm prototype contains all

descriptor calculation algorithms of priorities A and B. OpenBabel has not been prototyped yet as a separate software component, but it is implicitly used in the Fminer descriptor calculation implementation for substructure matching.

Classification and regression algorithms: These services are responsible for the generation of QSAR models which are stored on the server side and can be used for predicting toxicological properties. They require the specification of a dataset URI and of a URI for the prediction feature. Some algorithms accept additional parameters that allow the fine tuning of the training procedure.

For the first prototype we have implemented all algorithms of priority A (MLR as basic regression method, kNN as basic instance-based lazy learning classification method, J48 decision trees as an eager classification algorithm implementation and PLS which is very prominent in the QSAR community). Many algorithms of Priority B (SVM, Lazar, ToxTree) or even Priority C (Gaussian processes for regression, M5P, MaxTox) have also been prototyped. Additionally, we have implemented one popular machine learning classification algorithm that was not considered in Deliverable D4.1, namely the Naive Bayes classification method.

Clustering Algorithms: This category contains unsupervised learning algorithms that group objects of similar kind into respective categories. In other words clustering algorithms are exploratory data analysis tools which aim at sorting different objects into groups in a way that the degree of association between two objects is maximal if they belong to the same group and minimal otherwise. The most popular clustering algorithm, namely the k-means clustering method has been implemented in the initial prototype.

Feature Selection Algorithms: The fourth category contains algorithms for the reduction of the dimensionality of a dataset, by selecting only a subset of a full set of descriptors included in the dataset. The only feature selection algorithm of priority A (Info Gain Attribute Evaluation) has been included in the initial prototype.

It should be noted that all algorithm prototypes follow common OpenTox APIs[15], regardless of the category to which they belong. Each algorithm that has been prototyped is described in the next section of this report by a short text description and a detailed description of the web services that have been developed to implement the algorithm. Each algorithm implementation is presented in a uniform tabular format that has five logical parts, described below.

4.1 General Information about the service

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

4.2 Request/Response information

All algorithm web services require a number of input parameters and respond by providing the URI of the produced resource (model URI, datasets URI or feature URI). Some of the input parameters are mandatory. Besides the mandatory parameters, additional algorithm-specific parameters may need to be specified for some services. This second part of the table presents and describes all the input information and the results that are associated with each web service implementation.

[15] www.opentox.org/dev/apis/api-1.1/Algorithm

4.3 Status codes

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

4.4 Implementation Information

The fourth part of the table provides technical details about the prototype implementation, such as the type of HTTP method that is used to execute the service, the programming languages and the open source libraries that were integrated into the service.

4.5 Examples

The last part of the table provides examples of the web service. The examples are based on the cURL command; they can be easily invoked from the command line and can be used by technical reviewers to test and evaluate the performance of each service.

5. Prototype documentation

5.1 Descriptor calculation algorithms

5.1.1 FreeTreeMiner (TUM)

The FreeTreeMiner (FTM) software computes all acyclic substructures (in mathematical terms: *free* or *unrooted trees*) occurring at a given minimum frequency in a set of molecules. The substructures are computed by a depth-first search. Additionally to the minimum frequency support, a maximum frequency constraint can be set. This constraint can either refer to the same database/set or to a second one, meaning that all substructures frequent in the first and infrequent in the second are returned by FTM. The frequent substructures are returned as SMARTS strings together with their occurrences in the given set of structures.

General Information about the service
Service description The FreeTreeMiner (FTM) software computes all acyclic substructures occurring at a given minimum frequency in a set of molecules.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/FTM
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed Initial version completed in Feb. 2010
Partner responsible for the implementation Technische Universität München
Contact within OT Email: kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri= http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/39 which should be available in RDF format. This is a mandatory parameter. **minSup** All free trees that exceed the minimum support (minSup) are calculated.

	hydrogen Determines whether hydrogen atoms are taken into account.
Response	Once the features for a dataset are generated successfully, datasets URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/ftm is not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/FTM. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you have posted are acceptable but some internal error has occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information	
HTTP Method	POST
Programming Language	The project was built in Java and runs as a standalone application.
Libraries used	FTM software package, Open Babel

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uniplovdiv.bg:8080/ambit2/dataset/157" -d "minSup=0.9" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/FTM

5.1.2 FMiner (IST)

Fminer is a novel method for efficiently mining relevant tree-shaped subgraph descriptors with minimum frequency and correlation constraints, each representing a set of fragments sharing a common core structure (backbone), thereby reducing feature set size and runtime. The approach is able to optimize structural inter-feature entropy as opposed to occurrences, which is characteristic for open or closed fragment mining. In the experiments, the proposed method reduces feature set sizes by >90% and >30% compared to complete tree mining and open tree mining, respectively. Evaluation using cross-validation runs shows that their classification accuracy is similar to the complete set of trees but significantly better than that of open trees. Compared to open or closed fragment mining, a large part of the search space can be pruned due to an improved statistical constraint (dynamic upper bound adjustment), which is also confirmed in the experiments in lower runtimes compared to ordinary (static) upper bound pruning. Further analysis using large-scale datasets yields insight into important properties of the proposed descriptors, such as dataset coverage and class size represented by each descriptor. In a large-scale experiment, it was shown that the novel descriptors render large training sets feasible which previously might have been intractable for computational models.

General Information about the service
Service description An OpenTox REST Webservice Implements the OpenTox algorithm API for fminer
URI webservices.in-silico.ch/test/algorithm
OpenTox API Reference http://www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 1
Partner responsible for the implementation In Silico Toxicology (IST)
Contact within OT Christoph Helma helma@in-silico.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri:** is mandatory for all kind of prediction algorithms (machine learning or otherwise), as well for data processing algorithms. (webservices.in-silico.ch/test/dataset) **feature_uri:** is mandatory for prediction (classification/regression) and other supervised learning algorithms. The URI of the feature with the endpoint to predict is expected as value. **feature_generation_uri:** URI to the executing algorithm (e.g. http://webservices.in-silico.ch/test/algorithm/fminer). Supported MIME formats (chemical-mime.sourceforge.net/): application/rdf+xml (default): read/write OWL-DL
Response Once the model/features are successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and a new model was generated. The URI of the model is returned within the response body.
400	Bad request (Syntax error)
404	Not found (Resource not available)
500	Internal Server Error (request failed, e.g. prediction error)
503	Service Unavailable

Implementation Information
HTTP Method POST
Programming Language The project was built in Ruby and runs as a standalone application. Online CVSs are available at: github.com/helma/opentox-algorithm
Libraries used openbabel: openbabel.org/wiki/Main_Page fminer: www.maunz.de/libfminer-doc/

Examples
Example 1 Get the OWL-DL representation of fminer curl http://webservices.in-silico.ch/test/algorithm/fminer
Example 2 Create fminer features curl -X POST -d dataset_uri={datset_uri} -d feature_uri={feature_uri} http://webservices.in-silico.ch/test/algorithm/fminer (feaure_uri specifies the dependent variable, e.g. http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome_CPDBAS_Hamster) Creates a dataset with fminer features (backbone refinement class representatives from supervised graph mining, see http://www.maunz.de/libfminer-doc/). These features can be used e.g. as structural alerts, as descriptors (fingerprints) for prediction models or for similarity calculations.

5.1.3 gSpan' (TUM)

The gSpan' algorithm implements two optimizations of the widely known gSpan algorithm for mining molecular databases. Both optimizations apply to the enumeration of subgraph occurrences in a graph database, which is, also according to our profiling, the most expensive operation of gSpan. The first optimization reduces the number of subgraph isomorphisms that need to be accessed for proper support computation in considering the symmetries inherent in many chemical molecules, and the second speeds up subgraph isomorphism tests by making use of the non-uniform frequency distribution of atom and bond types.

General Information about the service
Service description The gSpan' software is able to compute all paths, trees and graphs that occur at a given minimum frequency in a set of molecules.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/gSpan
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed Initial version completed
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter. **minSup** All free trees that are equal to or exceed the minimum support (minSup) are calculated. **embeddingLists** Use embedding lists. **symmetries** Use symmetries. **fragmentsWithMaxEdges** Restrict search to fragments with maximum i edges. **NumMaxEdges** Number of maximum i edges. (Feature corresponds to fragmentsWithMaxEdges; it can only be set if fragmentsWithMaxEdges is set to 1). **linearFragments** Restrict search to linear fragments. **acyclicFragments**

	Restrict search to acyclic fragments.
Response	
	Once the features for a dataset are generated successfully, datasets URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	
	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
400	
	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	
	The resource was not found – Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/gspan is not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/gSpan. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	
	Internal Server Error – The parameters you posted are acceptable but some internal error occurred.
502	
	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down
503	
	The service is not available for the time being – Try again later!

Implementation Information	
HTTP Method	
	POST
Programming Language	
	The project was built in Java and runs as a standalone application.
Libraries used	
	gSpan' software package

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/157" -d "minSup=17" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/gSpan

5.1.4 JOELIB2

JOELIB2 is a platform-independent open source computational chemistry package written in Java. JOELIB2 consists of an algorithm library that was designed for prototyping, data mining and graph mining of chemical compounds. JOELib2 is the Java successor of the OELib library from OpenEye.

General Information about the service
Service description The JOELIB2 web service enables the user to calculate physicochemical, geometrical descriptors, functional groups, atom properties and fingerprints.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/JOELIB2
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 09
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter. **ALL** Calculate all available descriptors. **AcidicGroups**

Number of acidic groups.

AliphaticOHGroups

Number of aliphatic hydroxy groups.

AromaticOHGroups

Number of aromatic hydroxy groups.

AromaticBonds

Number of aromatic bonds.

BasicGroups

Number of basic groups.

FractionRotatableBonds

Fraction of rotatable bonds.

GeometricalDiameter

Calculates the geometrical diameter.

GeometricalRadius

Calculates the geometrical radius.

GeometricalShapeCoefficient

Calculates the geometrical shape coefficient.

GlobalTopologicalChargeIndex

Calculates the Topological Charge Index.

GraphShapeCoefficient

Calculates the graph shape coefficient.

HeavyBonds

Number of heavy bonds.

HeteroCycles

Number of hetero cycles.

HydrophobicGroups

Number of hydrophobic groups.

KierShape1

Calculates the Kier Shape for paths with length one.

KierShape2

Calculates the Kier Shape for paths with length two.

KierShape3

Calculates the Kier Shape for paths with length three.

LogP

Calculates the Octanol/Water partition coefficient (logP) or hydrophobicity.

MolarRefractivity

Calculates the molar refractivity (MR).

MolecularWeight

Calculates the molecular weight.

NumberOfAtoms

Number of atoms.

NumberOfBr

Number of bromium atoms.

NumberOfBonds

Number of bonds.

NumberOfC

Number of carbon atoms.

NumberOfCl

Number of chlorine atoms.

NumberOfHal

Number of halogen atoms.

NumberOfI

Number of iod atoms.

NumberOfF

Number of fluorine atoms.

NumberOfN

Number of nitrogen atoms.

NumberOfO

Number of oxygen atoms.

NumberOfP

Number of phospor atoms.

NumberOfS

Number of sulfur atoms.

NO2Groups

Number of NO2 groups.

OSOGroups

Number of OSO groups.

RotatableBonds

Number of rotatable bonds.

SOGroups

	Number of SO2 groups. **SO2Groups** Number of SO groups. **TopologicalDiameter** Calculates the topological diameter. **TopologicalRadius** Calculates the topological radius. **SSKey3DS** Pharmacophore fingerprint.
Response	
	Once the features for a dataset are generated successfully, datasets URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/joelib2is not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/JOELIB2. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you posted are acceptable but some internal error has occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application.
Libraries used The service uses the JOELIB2 software package.

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/157" -d "NumberOfF=true" -d "KierShape3=true" -d "FractionRotatableBonds=true" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/JOELIB2

5.1.5 The Chemistry Development Kit (CDK)

The Chemistry Development Kit (CDK) is a Java library for structural chemo- and bioinformatics. It is now developed by more than 50 developers all over the world and used in more than 10 different academic as well as industrial projects world wide. A number of descriptor implementations are available.

General Information about the service
Service description The CDK web service enables the user to calculate a variety of physicochemical and other descriptors.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/CDKPhysChem
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 09
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information

Posted Parameters

dataset_uri

(e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter.

ALL

Calculate all available descriptors.

ELECTRONIC

Calculate all available electronic descriptors.

GEOMETRICAL

Calculate all available geometrical descriptors.

CONSTITUTIONAL

Calculate all available constitutional descriptors.

HYBRID

Calculate all available hybrid descriptors.

TOPOLOGICAL

Calculate all available topological descriptors.

ALOGP

Calculates atom additive logP and molar refractivity values as described by Ghose and Crippen.

APol

Descriptor that calculates the sum of the atomic polarizabilities (including implicit hydrogens).

AminoAcidCount

Returns the number of amino acids found in the system.

AromaticAtomsCount

Descriptor based on the number of aromatic atoms of a molecule.

AromaticBondsCount

Descriptor based on the number of aromatic bonds of a molecule.

AtomCount

Descriptor based on the number of atoms of a certain element type.

AutocorrelationCharge

The Moreau-Broto autocorrelation descriptors using partial charges.

AutocorrelationMass

The Moreau-Broto autocorrelation descriptors using atomic weight.

AutocorrelationPolarizability

The Moreau-Broto autocorrelation descriptors using polarizability.

BCUT

Eigenvalue-based descriptor noted for its utility in chemical diversity described by Pearlman et al.

BPol

Descriptor that calculates the sum of the absolute value of the difference between atomic polarizabilities of all bonded atoms in the molecule (including implicit hydrogens).

BondCount

Descriptor based on the number of bonds of a certain bond order.

CPSA

A variety of descriptors combining surface area and partial charge information.

CarbonTypes

Characterizes the carbon connectivity in terms of hybridization.

ChiChain

Evaluates the Kier & Hall Chi chain indices of orders 3,4,5 and 6.

ChiCluster

Evaluates the Kier & Hall Chi cluster indices of orders 3,4,5,6 and 7.

ChiPathCluster

Evaluates the Kier & Hall Chi path cluster indices of orders 4,5 and 6.

ChiPath

Evaluates the Kier & Hall Chi path indices of orders 0,1,2,3,4,5,6 and 7.

EccentricConnectivityIndex

A topological descriptor combining distance and adjacency information.

FragmentComplexity

Class that returns the complexity of a system. The complexity is defined as @cdk.cite{Nilakantan06}.

GravitationalIndex

Descriptor characterizing the mass distribution of the molecule.

HBondAcceptorCount

Descriptor that calculates the number of hydrogen bond acceptors.

HBondDonorCount

Descriptor that calculates the number of hydrogen bond donors.

IPMolecularLearning

Descriptor that evaluates the ionization potential.

KappaShapeIndices

Descriptor that calculates Kier and Hall kappa molecular shape indices.

KierHallSmarts

Counts the number of occurrences of the E-state fragments.

LargestChain

Returns the number of atoms in the largest chain.

LargestPiSystem

Returns the number of atoms in the largest pi chain.

LengthOverBreadth

Calculates the ratio of length to breadth.

LongestAliphaticChain

Returns the number of atoms in the longest aliphatic chain.

MDE

Evaluates molecular distance edge descriptors for C, N and O.

MomentOfInertia

Descriptor that calculates the principal moments of inertia and ratios of the principal moments. Als calculates the radius of gyration.

PetitjeanNumber

Descriptor that calculates the Petitjean Number of a molecule.

PetitjeanShapeIndex

The topological and geometric shape indices described by Petitjean and Bath et al. respectively. Both measure the anisotropy in a molecule.

RotatableBondsCount

Descriptor that calculates the number of non-rotatable bonds on a molecule.

RuleOfFive

This Class contains a method that returns the number of failures of the Lipinski's Rule Of Five.

TPSA

Calculation of topological polar surface area based on fragment contributions.

VAdjMa

Descriptor that calculates the vertex adjacency information of a molecule.

WHIM

Holistic descriptors described by Todeschini et al.

Weight

Descriptor based on the weight of atoms of a certain element type. If no element is specified, the returned value is the Molecular Weight.

WeightedPath

The weighted path (molecular ID) descriptors described by Randic. They characterize molecular branching.

WienerNumbers

	This class calculates Wiener path number and Wiener polarity number. XLogP Prediction of logP based on the atom-type method called XLogP. ZagrebIndex The sum of the squared atom degrees of all heavy atoms.
Response	Once the features for a dataset are generated successfully, datasets URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/cdkphyschem is not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/CDKPhysChem. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you have posted are acceptable but some internal error has occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information	
HTTP Method	POST
Programming Language	The project was built in Java and runs as a standalone application.
Libraries used	The service uses the CDK software package.

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/157" -d "HBondAcceptorCountDescriptor=true" -d "ALL=false" -d "ELECTRONIC=true" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/CDKPhysChem

5.1.6 MakeMNA/MakeQNA

MakeMNA is a software product for generating MNA descriptors.

These descriptors are based on the molecular structure representation, which includes the hydrogens according to the valences and partial charges of other atoms and does not specify the types of bonds. MNA descriptors are generated as a recursively defined sequence:

- zero-level MNA descriptor for each atom is the mark A of the atom itself;
- any next-level MNA descriptor for the atom is the sub-structure notation A(D1,D2,..Di…), where Di is the previous-level MNA descriptor for i-th immediate neighbors of the atom A.

The mark of atom may include not only the atomic type but also any additional information about the atom. In particular, if the atom is not included into the ring, it is marked by "-". The neighbor descriptors D1,D2,...Di… are arranged in a unique manner, e.g., in lexicographic order. An iterative process of MNA descriptors generation can be continued covering first, second, etc. neighborhoods of each atom.

MakeQNA is a software product for generating QNA descriptors.

Quantitative Neighborhoods of Atoms (QNA) descriptors are based on quantities of ionization potential (IP) and electron affinity (EA) of each atom of the molecule. They are calculated as follows:

- Pi = Bi-½Σk(exp(-½C))ikBk-½,
- Qi = Bi-½Σk(exp(-½C))ikBk-½Ak,
- Ai = ½(IPi + EAi), Bi = IPi - EAi.

Where IPi is the ionization potential (the energy required to remove the outermost electron from a neutral gaseous atom), and EAi is the electron affinity (the energy released when an electron is added to a neutral gaseous atom of that element) of atom i.

General Information about the service
Service description The MakeMNA/MakeQNA web service provides the possibility to calculate Multilevel Neighborhoods of Atom (MNA) descriptors and Quantitative Neighborhoods of Atoms (QNA) descriptors for structures of chemicals represented in MOL V2000 format (SYMYX MDL).
URI 195.178.207.160/OpenTox/algorithm/MakeMNA
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm

Date completed	
	2010 – February 15
Partner responsible for the implementation	
	Institute of Biomedical Chemistry of Russian Academy of Medical Sciences
Contact within OT	
	Alexey Zakharov alexey.zakharov@ibmc.msk.ru
Comments (including reviews)	

Request/Response Information	
Posted Parameters	
	dataset_uri (e.g. dataset_uri= 195.178.207.160/OpenTox/dataset/3) which should be available in RDF format. This is a mandatory parameter.
Response	
	Once the feature is successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and a new feature was generated. The URI of the model is returned within the response body.
400	Bad Request – Some parameter you provided is wrong or you didn't post some mandatory parameter such as the dataset_uri or/and the target. A list of errors and explanatory messages is returned within the response.
404	The resource was not found – Check your spelling. For a complete list of all available algorithms, check out http://195.178.207.160:7000/algorithm/MNA http://195.178.207.160:7000/algorithm/QNA
500	Internal Server Error – The parameters you posted are acceptable but some internal error occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Delphi/PHP and runs as a standalone service.
Libraries used None

Examples
Example 1 curl -X POST -d "dataset_uri=http://195.178.207.160/OpenTox/dataset/3" http://195.178.207.160/OpenTox/algorithm/MakeMNA

5.2 Classification and regression algorithms

5.2.1 Gaussian processes for regression

GPR (Gaussian Processes for Regression) is a method of supervised learning. A Gaussian process is a generalization of the Gaussian probability distribution. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic process governs the properties of functions [RAS05]. Just as a Gaussian distribution is fully specified by its mean and covariance matrix, a Gaussian process is specified by a mean and covariance function. Here, the mean is a function of x (which we will often take to be the zero function), and the covariance is a function C(x, x') that expresses the expected covariance between the values of the function y at the points x and x'. The function y(x) in any one data modeling problem is assumed to be a single sample from this Gaussian distribution.

General Information about the service
Service description The Gaussian Process web service enables the user to build regression models for a specific dataset.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 09

Partner responsible for the implementation
Technische Universität München
Contact within OT
kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters
dataset_uri
(e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter.
kernel
{PolyKernel, Puk, RBFKernel}
the kernel to use
gamma
parameter for the rbf kernel only
omega
parameter for the puk kernel only
sigma
parameter for the puk kernel only
exponent
parameter for the polynomial kernel only
noise
parameter for the polynomial kernel only
Response
A task URI is provided if the service tries to calculate a regression model. The task URI can be queried (GET) for the resulting model URI. Otherwise an explanatory message is provided.

Status Codes
200
Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.

303	Redirect – the result can be found elsewhere.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/gaussp not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you posted are acceptable but some internal error has occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application.
Libraries used The service uses the WEKA software package.

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/23" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/12818" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP
Example 2 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/158" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/29634" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP

5.2.2 MLR

MLR (Multiple Linear Regression) is a simple and popular statistical technique that uses several explanatory (independent) variables to predict the outcome of a response (dependent) variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points. The algorithm type according to the OpenTox ontology is: Algorithm-MLDMTox-Learning-Regression-eager.

General Information about the service
Service description Train a multiple linear regression model providing a dataset URI, the target feature of the dataset. No tuning parameters are required for this algorithm. MLR models express the dependent variable as a linear combination of the other variables plus a bias (intercept). Once the model is successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.
URI opentox.ntua.gr:3000/algorithm/mlr
Date completed 2009 - December 28
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Partner responsible for the implementation National Technical University of Athens
Contact within OT Pantelis Sopasakis chung@central.ntua.gr
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=opentox.ntua.gr/ds.rdf) which should be available in RDF format. This is a mandatory parameter. **Target** The URI of the feature of the dataset that should be used as the target (dependent) variable while training the model. This should not only be a valid URI but additionally be a feature of the dataset. This is a mandatory parameter.
Response Once the model is successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes
200 Success – The request has succeeded and a new model was generated. The URI of the model is returned within the response body.
400 Bad Request – Some parameter you provided is wrong or you didn't post some mandatory parameter such as the dataset_uri or/and the target. A list of errors and explanatory messages is returned within the response.
404 The following message is returned to the client: You have requested an algorithm which does not exist. You can get a complete list of all available algorithms at opentox.ntua.gr:3000/algorithm
500 Internal Server Error – The parameters you posted are acceptable but some internal error occurred.
502 Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503 The service is not available for the time being – Try again later!

Implementation Information
HTTP Method
POST
Programming Language
Java
Libraries used
The service uses the class LinearRegression which is included in Weka (version 3.6.0)

Examples
Example 1. Train an MLR Regression Model
curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/6" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/11954" http://opentox.ntua.gr:3000/algorithm/mlr

5.2.3 SVM

Support vector machines (SVM) are a set of supervised learning methods used for classification and regression. In the most widely used two-class SVM classification method, input data are viewed as two sets of vectors in the multi-dimensional input space. The SVM classifier constructs a separating hyperplane in that space, one which maximizes the margin between the two data sets. The method is extended to multi-class and nonlinear classification problems by using a nonlinear kernel function. To obtain an optimum classifier for non-separable data, a penalty is introduced for misclassified data. This penalty is zero for patterns classified correctly, and has a positive value that increases with the distance from the corresponding hyperplane for patterns that are not situated on the correct side of the classifier. Similar concepts are used in the SVM regression problem, where the objective is to identify a function that for all training patterns has a maximum deviation ϵ from the target (experimental) values. The algorithm type according to the OpenTox ontology is: Algorithm-MLDMTox-Learning-Classification-eager for the SVM classification method and Algorithm-MLDMTox-Learning-Regression-eager for SVM regression

5.2.3.1 SVM regression

General Information about the service
Service description
Train an SVM regression model providing a dataset URI, the target feature of the dataset and optionally a set of tuning parameters such as the kernel to be used.
URI
opentox.ntua.gr:3000/algorithm/svm

Date completed	
	2009 - December 28
OpenTox API Reference	
	www.opentox.org/dev/apis/api-1.1/Algorithm
Partner responsible for the implementation	
	National Technical University of Athens
Contact within OT	
	Pantelis Sopasakis chung@central.ntua.gr
Comments (including reviews)	

Request/Response Information

Posted Parameters

dataset_uri

(e.g. dataset_uri=opentox.ntua.gr/ds.rdf) which should be available in RDF format. This is a mandatory parameter.

Target

The URI of the feature of the dataset that should be used as the target (dependent) variable while training the model. This should not only be a valid URI but additionally be a feature of the dataset. This is a mandatory parameter.

Kernel

Is the kernel function of the Support Vectors. You may choose among rbf, linear and polynomial. This parameter is not case sensitive, so kernel=rbf and kernel=RBF is the same. This parameter is optional and its default value is RBF. If you use the RBF kernel you may specify the gamma parameter, while if you choose the polynomial one, you may specify the degree and coeff0 parameters.

Gamma

The γ parameter for the RBF kernel. This is an optional parameter and its default value is 1.0. The value of gamma you specify must be double and strictly positive, otherwise an error message is returned and the status code of the response is set to 400 (Bad request). This parameter will be omitted if combined with kernels other than RBF.

Degree

When using the Polynomial kernel, degree is the maximum exponent of the kernel function. This parameter has to be integer and greater or equal to 1, otherwise an error message is returned and the status code of the response is set to 400 (Bad request). This is an optional parameter and its default value is 3.

Epsilon

The ϵ - parameter of the ϵ-SVR algorithm that is used for the training. The parameter is optional and its default value is 0.1. epsilon can be any positive double.

cacheSize

Size of cache to be used for storing the SVM kernels in bytes. Must be a non-negative integer and

prime. This parameter is optional and its default value is 250007. If set to 0, all available cache will be used. For no cache, set this variable to -1.

Tolerance

Tolerance used as a convergence/termination criterion while training the model. Large values of tolerance imply fast computations but less accurate models. Extremely small tolerance values (e.g. 1E-12) should be avoided as they could impose great computational burdens and destabilize the convergence of the training algorithm. The default value for this parameter is 0.0001 or 1E-4.

coeff0

It is the bias of the kernel k(x,y)+coeff0

Cost

Coefficient of the cost function used to train the model. This is a strictly positive double valued parameter which is optional and its default value is 10.0.

Response

Once the model is successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and a new model was generated. The URI of the model is returned within the response body.
400	Bad Request – Some parameter you provided is wrong or you didn't post some mandatory parameter such as the dataset_uri or/and the target. A list of errors and explanatory messages is returned within the response.
404	The following message is returned to the client: You have requested an algorithm which does not exist. You can get a complete list of all available algorithms at opentox.ntua.gr:3000/algorithm
500	Internal Server Error – The parameters you posted are acceptable but some internal error occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application. Online CVSs are available at: github.com/alphaville/yaqp-turbo
Libraries used The service uses the class SVMreg which is included in Weka (version 3.6.0)

Examples
Example 1 – Train an SVM Regression Model using the default parameter values curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/6" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/11954" http://opentox.ntua.gr:3000/algorithm/svm
Example 2 – Train an SVM Regression Model by setting the parameter values curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/6" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/11954&gamma=1.5&epsilon=0.034&cacheSize=1000&tolerance=0.01&kernel=RBF&cost=1000" http://opentox.ntua.gr:3000/algorithm/svm

5.2.3.2 SVM classification

General Information about the service
Service description Train an SVM classification model providing a dataset URI, the target feature of the dataset and optionally a set of tuning parameters such as the kernel to be used. The service can be applied on datasets that include nominal features.
URI opentox.ntua.gr:3000/algorithm/svm
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm

Date completed
2009 - December 28
Partner responsible for the implementation
National Technical University of Athens
Contact within OT
Pantelis Sopasakis chung@central.ntua.gr
Comments (including reviews)

Request/Response Information

Posted Parameters

dataset_uri

(e.g. dataset_uri=opentox.ntua.gr/ds.rdf) which should be available in RDF format. This is a mandatory parameter. Note that the training dataset must have at least one nominal feature. All string features are excluded from the training procedure.

Target

The URI of the feature of the dataset that should be used as the target (dependent) variable while training the model. This should not only be a valid URI but additionally be a feature of the dataset and must be nominal. This is a mandatory parameter.

Kernel

Is the kernel function of the Support Vectors. You may choose among rbf, linear and polynomial. This parameter is not case sensitive, so kernel=rbf and kernel=RBF is the same. This parameter is optional and its default value is RBF. If you use the RBF kernel you may specify the gamma parameter while if you choose the polynomial one, you may specify the degree and coeff0 parameters.

Gamma

The γ parameter for the RBF kernel. This is an optional parameter and its default value is 1.0. The value of gamma you specify must be double and strictly positive, otherwise an error message is returned and the status code of the response is set to 400 (Bad request). This parameter will be omitted if combined with kernels other than RBF.

Degree

When using the Polynomial kernel, degree is the maximum exponent of the kernel function. This parameter has to be integer and greater or equal to 1, otherwise an error message is returned and the status code of the response is set to 400 (Bad request). This is an optional parameter and its default value is 3.

cacheSize

Size of cache to be used for storing the SVM kernels in bytes. Must be a non-negative integer and prime. This parameter is optional and its default value is 250007. If set to 0, all available cache will be used. For no cache, set this variable to -1.

Tolerance

	Tolerance used as a convergence/termination criterion while training the model. Large values of tolerance imply fast computations but less accurate models. Extremely small tolerance values (e.g. 1E-12) should be avoided as they could impose great computational burdens and destabilize the convergence of the training algorithm. The default value for this parameter is 0.0001 or 1E-4. **coeff0** It is the bias of the kernel k(x,y)+coeff0 **Cost** Coefficient of the cost function used to train the model. This is a strictly positive double valued parameter which is optional and its default value is 10.0.
Response	
	Once the model is successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	
	Success – The request has succeeded and a new model was generated. The URI of the model is returned within the response body.
400	
	Bad Request – Some parameter you provided is wrong or you didn't post some mandatory parameter such as the dataset_uri or/and the target. A list of errors and explanatory messages is returned within the response.
404	
	The following message is returned to the client: You have requested an algorithm which does not exist. You can get a complete list of all available algorithms at opentox.ntua.gr:3000/algorithm
500	
	Internal Server Error – The parameters you posted are acceptable but some internal error occurred.
502	
	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	
	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application. Online CVSs are available at: github.com/alphaville/yaqp-turbo
Libraries used The service uses the class SMO of Weka.

Examples
Example 1 – Train an SVM Classification Model using the default parameter values curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/9" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/12136" http://opentox.ntua.gr:3000/algorithm/svc
Example 2 – Train an SVM Classification Model by setting the parameter values curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/9" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/12136&gamma=1.5&cacheSize=1000&tolerance=0.01&kernel=RBF&cost=1000" http://opentox.ntua.gr:3000/algorithm/svc
Example 3 – Use an SVM model for prediction Let http://opentox.ntua.gr:3000/model/x be a model produced by the abovementioned service (examples 1 and 2). Then the following command can be used to obtain predictions for a given dataset: curl -X POST -d 'dataset_uri=http://abmit.uni-plovdiv.bg:8080/ambit2/dataset/9' http://opentox.ntua.gr:3000/model/x The produced dataset contains the data entries of the initial dataset plus the predicted ones.
Example 4 – Get the predicted, independent and dependent features from a model resource curl http://opentox.ntua.gr:3000/model/model_id/predicted curl http://opentox.ntua.gr:3000/model/model_id/dependent curl http://opentox.ntua.gr:3000/model/model_id/independent

5.2.4 KNN

The k-nearest neighbors algorithm (kNN) is a method for classifying objects based on closest training examples in the feature space. It is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is delayed until classification. A majority vote of an object's neighbors is used for classification, with the object being assigned to the class most common amongst its k

(positive integer, typically small) nearest neighbors. If k is set to 1, then the object is simply assigned to the class of its nearest neighbor. The knn algorithm can also be applied for regression in the same way by simply assigning the property value for the object to be the average of the values of its k nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. No explicit training step is required since training consists of just storing training instance feature vectors and corresponding class labels. In order to identify neighbors, the objects are represented by position vectors in a multidimensional feature space. It is usual to use the Euclidean distance, but also further distance measures, such as the Manhattan distance could be used instead. In the classification/testing phase, the test sample is represented as a vector in the feature space. Distances from this vector to all stored vectors are computed and the k closest samples are selected to determine the class/real-value of the test instance.
The k-nearest neighbor algorithm is sensitive to the local structure of the data. The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques like cross-validation. The accuracy of the kNN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance.

General Information about the service
Service description The kNNregression and kNNclassification web services enable the user to build regression/classification models for a specific dataset using the lazy nearest neighbor approach.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNregression opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNclassification
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 09
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter.

	KNN set the k parameter - number of neighbors considered. **distanceWeighting** 0 for no distance weighting, I for 1/distance or F for 1-distance. **meanSquared** Whether the mean squared error is used rather than mean absolute error when doing cross-validation for regression problems. **windowSize** Gets the maximum number of instances allowed in the training pool. The addition of new instances above this value will result in old instances being removed. A value of 0 signifies no limit to the number of training instances.
Response	A task URI is provided if the service tries to calculate a regression model. The task URI can be queried (GET) for the resulting model URI. Otherwise an explanatory message is provided.

Status Codes	
200	Success - The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
303	Redirect - the result can be found elsewhere.
400	Bad Request - Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found - Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/knnRegression not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNregression. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error - The parameters you posted have are acceptable but some internal error has occurred.
502	Bad Gateway - The service was unsuccessful because while the server was acting as a client, it

	received an unsuccessful response. In such a case, it seems that some other server is down.
503	
	The service is not available for the time being - Try again later!

Implementation Information
HTTP Method
POST
Programming Language
The project was built in Java and runs as a standalone application.
Libraries used
The service uses the WEKA software package.

Examples
Example 1
curl -X POST -d "dataset_uri=http://ambit.uniplovdiv.bg:8080/ambit2/dataset/23" -d "prediction_feature=http://ambit.uniplovdiv.bg:8080/ambit2/feature/12818" -d 'KNN=7' http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNregression
Example 2
curl -X POST -d "dataset_uri=http://ambit.uniplovdiv.bg:8080/ambit2/dataset/23" -d "prediction_feature=http://ambit.uniplovdiv.bg:8080/ambit2/feature/12818" -d 'KNN=7' http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNclassification

5.2.5 Lazar (IST)

Lazar is a k-nearest-neighbor approach to predict chemical endpoints from a training set based on structural fragments. It uses pre-computed fragments with occurrences as well as target class information for each compound as training input. It also features regression, in which case the target activities consist of continuous values. Lazar uses activity-specific similarity (i.e. each fragment contributes with its significance for the target activity) that is the basis for predictions and confidence index for every single prediction.

For classification, a weighted nearest neighbor voting is the standard prediction, whereas for regression a kernel model based on activity-specific similarity is used by default. A kernel model is also available for classification, as well as a multi-linear model for regression.

General Information about the service
Service description
An OpenTox REST Webservice that implements the OpenTox algorithm API for lazar.

URI
webservices.in-silico.ch/test/algorithm
OpenTox API Reference
www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed
2010 - February 1
Partner responsible for the implementation
In Silico Toxicology (IST)
Contact within OT
Christoph Helma helma@in-silico.de
Comments (including reviews)

Request/Response Information

Posted Parameters

dataset_uri:

is mandatory for all kind of prediction algorithms (machine learning or otherwise), as well for data processing algorithms.

(webservices.in-silico.ch/test/dataset)

feature_uri:

is mandatory for prediction (classification/regression) and other supervised learning algorithms. The URI of the feature with the endpoint to predict is expected as value.

feature_generation_uri:

URI to the executing algorithm (e.g. http://webservices.in-silico.ch/test/algorithm/fminer).

Supported MIME formats (chemical-mime.sourceforge.net/):

application/rdf+xml (default): read/write OWL-DL

Response

Once the model/features are successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes

200

Success - The request has succeeded and a new model was generated. The URI of the model is returned within the response body.

400	
	Bad request (Syntax error)
404	
	Not found (Resource not available)
500	
	Internal Server Error (request failed, e.g. prediction error)
503	
	Service Unavailable

Implementation Information
HTTP Method
GET, POST
Programming Language
The project was built in Ruby and runs as a standalone application. Online CVSs are available at: github.com/helma/opentox-algorithm
Libraries used
openbabel: openbabel.org/wiki/Main_Page fminer: www.maunz.de/libfminer-doc/

Examples
Example 1
Get the OWL-DL representation of lazar curl http://webservices.in-silico.ch/test/algorithm/lazar
Example 2
Create lazar model curl -X POST -d dataset_uri={datset_uri} -d feature_uri={feature_uri} -d feature_generation_uri=http://webservices.in-silico.ch/test/algorithm/fminer http://webservices.in-silico.ch/test/algorithm/lazar (feaure_uri specifies the dependent variable, e.g. http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome_CPDBAS_Hamster)

5.2.6 J48

J48 implements Quinlan's C4.5 algorithm for generating a pruned or un-pruned C4.5 decision tree. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by J48 can be used for classification. J48 builds decision trees from a set of labeled training data using the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision by splitting the data into smaller

subsets. J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To make the decision, the attribute with the highest normalized information gain is used. Then the algorithm recurs on the smaller subsets. The splitting procedure stops if all instances in a subset belong to the same class. Then a leaf node is created in the decision tree telling to choose that class. But it can also happen that none of the features give any information gain. In this case J48 creates a decision node higher up in the tree using the expected value of the class.
J48 can handle both continuous and discrete attributes, training data with missing attribute values and attributes with differing costs. Further it provides an option for pruning trees after creation.

General Information about the service
Service description The J48 web service enables the user to build classification models for a specific dataset with the C4.5 algorithm.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/J48
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 09
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter. **binarySplits** use binary splits on nominal attributes when building trees. **confidenceFactor** the confidence factor used for pruning (smaller values incur more pruning). **minNumObj** minimum number of instances per leaf. **numFolds** Determines the amount of data used for reduced-error pruning. One fold is used for pruning, the

	rest for growing the tree. **reducedErrorPruning** Whether reduced-error pruning is used instead of C.4.5 pruning. **seed** The seed used for randomizing the data when reduced-error pruning is used. **subtreeRaising** Whether to consider the subtree raising operation when pruning. **unpruned** Whether pruning is performed. **useLaplace** Whether counts at leaves are smoothed based on Laplace.
Response	A task URI is provided if the service tries to calculate a regression model. The task URI can be queried (GET) for the resulting model URI. Otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
303	Redirect – the result can be found elsewhere.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you have posted are acceptable but some internal error has occurred.

502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application.
Libraries used The service uses the WEKA software package.

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uniplovdiv.bg:8080/ambit2/dataset/6" -d "prediction_feature=http://ambit.uniplovdiv.bg:8080/ambit2/feature/12818" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/J48

5.2.7 M5P

M5P is a reconstruction of Quinlan's M5 algorithm for inducing trees of regression models. M5P combines a conventional decision tree with the possibility of linear regression functions at the nodes. First, a decision-tree induction algorithm is used to build a tree, but instead of maximizing the information gain at each inner node, a splitting criterion is used that minimizes the intra-subset variation in the class values down each branch. The splitting procedure in M5P stops if the class values of all instances that reach a node vary very slightly, or only a few instances remain. Second, the tree is pruned back from each leaf. When pruning an inner node is turned into a leaf with a regression plane. Third, to avoid sharp discontinuities between the subtrees a smoothing procedure is applied that combines the leaf model prediction with each node along the path back to the root, smoothing it at each of these nodes by combining it with the value predicted by the linear model for that node. Techniques devised by Breiman et al. for their CART system are adapted in order to deal with enumerated attributes and missing values. All enumerated attributes are turned into binary variables so that all splits in M5P are binary. As to missing values, M5P uses a technique called "surrogate splitting" that finds another attribute to split on in place of the original one and uses it instead. During training, M5P uses as surrogate attribute the class value in the belief that this is the attribute most likely to be correlated with the one used for splitting. When the splitting procedure ends, all missing values are replaced by the average values of the corresponding attributes of the training examples reaching the leaves. During testing an unknown attribute value is replaced by the average value of that attribute for all training instances that reach the node, with the

effect of choosing always the most populated subnode.
M5P generates models that are compact and relatively comprehensible.

General Information about the service
Service description The M5P web service enables the user to build regression models for a specific dataset with the M5 algorithm by R. Quinlan and Yong Wang.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/M5P
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 09
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter. **binarySplits** use binary splits on nominal attributes when building trees. **confidenceFactor** the confidence factor used for pruning (smaller values incur more pruning). **minNumObj** minimum number of instances per leaf. **numFolds** Determines the amount of data used for reduced-error pruning. One fold is used for pruning, the rest for growing the tree, **reducedErrorPruning** Whether reduced-error pruning is used instead of C.4.5 pruning.

	seed The seed used for randomizing the data when reduced-error pruning is used. **subtreeRaising** Whether to consider the subtree raising operation when pruning. **unpruned** Whether pruning is performed. **useLaplace** Whether counts at leaves are smoothed based on Laplace.
Response	A task URI is provided if the service tries to calculate a regression model. The task URI can be queried (GET) for the resulting model URI. Otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
303	Redirect – the result can be found elsewhere.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you have posted are acceptable but some internal error has occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application.
Libraries used The service uses the WEKA software package.

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uniplovdiv.bg:8080/ambit2/dataset/39" -d "prediction_feature=http://ambit.uniplovdiv.bg:8080/ambit2/feature/12818" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/J48

5.2.8 Naive Bayes

The Naive Bayes algorithm is based on conditional probabilities. It uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data. Naive Bayes makes the assumption that each predictor is conditionally independent of the others. In practice, this assumption of independence, even when violated, does not degrade the model's predictive accuracy significantly, and makes the difference between a fast, computationally feasible algorithm and an intractable one.

General Information about the service
Service description One of the most simple yet widely used probabilistic models in classification theory is found in the naive Bayes approach. This is a web service which provides an interface for training such models. The service can be applied on datasets whose target variables are nominal.
URI opentox.ntua.gr:3000/algorithm/naiveBayes
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 22
Partner responsible for the implementation National Technical University of Athens

Contact within OT
Pantelis Sopasakis chung@central.ntua.gr
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=opentox.ntua.gr/ds.rdf) which should be available in RDF format. This is a mandatory parameter. Note that the training dataset must have at least one nominal feature. All string features are excluded from the training procedure. **prediction_feature** The URI of the feature of the dataset that should be used as the target (dependent) variable while training the model. This should not only be a valid URI but additionally be a feature of the dataset and must be nominal.
Response Once the model is successfully created, its URI is returned to the client within the response and the status is set to 200; otherwise an explanatory message is provided.

Status Codes
200 Success – The request has succeeded and a new model was generated. The URI of the model is returned within the response body.
400 Bad Request – Some parameter you provided is wrong or you didn't post some mandatory parameter such as the dataset_uri or/and the target. A list of errors and explanatory messages is returned within the response.
404 The following message is returned to the client: You have requested an algorithm which does not exist. You can get a complete list of all available algorithms at opentox.ntua.gr:3000/algorithm
500 Internal Server Error – The parameters you posted are acceptable but some internal error occurred.
502 Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503 The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application. Online CVSs are available at: github.com/alphaville/yaqp-turbo
Libraries used The service uses the class NaiveBayes of Weka.

Examples
Example 1 – Training curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/9" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/12136" http://opentox.ntua.gr:3000/algorithm/naiveBayes
Example 2 – Use a naive Bayes model for prediction Let http://opentox.ntua.gr:3000/model/x be a model produced by the above mentioned service. Then the following command can be used to obtain predictions for a given dataset: curl -X POST -d 'dataset_uri=http://abmit.uni-plovdiv.bg:8080/ambit2/dataset/9' http://opentox.ntua.gr:3000/model/x The dataset containing the predicted values is posted to an available dataset service (e.g. http://ambit.uni-plocdiv.bg:8080/ambit2/dataset)

5.2.9 ToxTree (IDEA)

Toxtree is a full-featured and flexible user-friendly open source application, which is able to estimate toxic hazard by applying a decision tree approach. Currently it includes the following modules:

1. Cramer rules
2. Verhaar scheme for predicting toxicity mode of actions
3. A decision tree for estimating skin irritation and corrosion potential.
4. A decision tree for estimating eye irritation and corrosion potential
5. A decision tree for estimating carcinogenicity and mutagenicity

Toxtree has been designed with flexible capabilities for future extensions in mind (e.g. other classification schemes that could be developed at a future date). New decision trees with arbitrary rules can be built with the help of a graphical user interface or by developing new plug-ins.

General Information about the service
Service description REST services, exposing Toxtree hazard estimation (toxtree.sourceforge.net/)
URI • Cramer rules apps.ideaconsult.net:8180/ambit2/algorithm/toxtreecramer • Extended Cramer rules apps.ideaconsult.net:8180/ambit2/algorithm/toxtreecramer2 • Eye irritation apps.ideaconsult.net:8180/ambit2/algorithm/toxtreeeye • Skin irritation apps.ideaconsult.net:8180/ambit2/algorithm/toxtreeskinirritation/ • Structure Alerts for the *in vivo* micronucleus assay in rodents apps.ideaconsult.net:8180/ambit2/algorithm/toxtreemic • Michael Acceptors by Structural Alerts apps.ideaconsult.net:8180/ambit2/algorithm/toxtreemichaelacceptors • Benigni/Bossa rules for carcinogenicity and mutagenicity apps.ideaconsult.net:8180/ambit2/algorithm/toxtreecarc • ILSI/Kroes decision tree for TTC apps.ideaconsult.net:8180/ambit2/algorithm/toxtreekroes
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm Relevant API: www.opentox.org/dev/apis/api-1.1/Model/ www.opentox.org/dev/apis/api-1.1/Task
Date completed 2009/12/21
Partner responsible for the implementation IDEA
Contact within OT jeliazkova.nina@gmail.com
Comments (including reviews)

Request/Response Information
Posted Parameters None
Response Returns Task URL (/task/{id}) or Model URL (/model/{id}) in HTTP Location header.

Status Codes
200 Success – The request has succeeded and a new model was generated. The URI of the model is returned in the HTTP Location: header.
202 Accepted – The request has been accepted, but model generation has not completed and a new model was generated. The URI of the task is returned in the HTTP Location: header.
303 Redirected (See Other) – The model generation is running. Returns Task URI in the HTTP Location: header.
404 Algorithm not found http://www.opentox.org/dev/apis/api-1.1/Algorithm
500 Internal server error

Implementation Information
HTTP Method POST
Programming Language Java ambit.svn.sourceforge.net/svnroot/ambit/trunk/ambit2-all/ambit2-www/src/main/java/ambit2/rest
Libraries used Multiple (Ambit, Toxtree, The Chemistry Development Kit, Restlet among them)

Examples
Example 1 : Create Toxtree model "Extended Cramer rules " curl -X POST http://apps.ideaconsult.net:8180/ambit2/algorithm/toxtreecramer2 -iv * About to connect() to apps.ideaconsult.net port 8180 (#0) * Trying 93.123.36.100... connected * Connected to apps.ideaconsult.net (93.123.36.100) port 8180 (#0) > POST /ambit2/algorithm/toxtreecramer2 HTTP/1.1 > User-Agent: curl/7.19.7 (amd64-portbld-freebsd8.0) libcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3 > Host: apps.ideaconsult.net:8180 > Accept: */* > < HTTP/1.1 303 See Other

```
< Server: Apache-Coyote/1.1
< Date: Sun, 14 Feb 2010 19:19:03 GMT
< Location: http://apps.ideaconsult.net:8180/ambit2/task/dfbd5f05-22f8-4b0d-ba8e-
b55918c9be90
< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
< Accept-Ranges: bytes
< Server: Restlet-Framework/2.0m6
< Content-Length: 0
<
* Connection #0 to host apps.ideaconsult.net left intact
* Closing connection #0
```

Example 2

Retrieving model URL from task url

```
curl -X GET http://apps.ideaconsult.net:8180/ambit2/task/50f8ec54-0b21-434b-9661-
8387d0c7d584 -iv

* About to connect() to apps.ideaconsult.net port 8180 (#0)
*   Trying 93.123.36.100... connected
* Connected to apps.ideaconsult.net (93.123.36.100) port 8180 (#0)
> GET /ambit2/task/50f8ec54-0b21-434b-9661-8387d0c7d584 HTTP/1.1
> User-Agent: curl/7.19.7 (amd64-portbld-freebsd8.0) libcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3
> Host: apps.ideaconsult.net:8180
> Accept: */*
>
< HTTP/1.1 303 See Other
< Server: Apache-Coyote/1.1
< Date: Sun, 14 Feb 2010 19:22:09 GMT
< Location: http://apps.ideaconsult.net:8180/ambit2/model/ToxTree%3A+Extended+Cramer+rules
< Accept-Ranges: bytes
< Server: Restlet-Framework/2.0m6
< Content-Length: 0
<
* Connection #0 to host apps.ideaconsult.net left intact
* Closing connection #0
```

Example 3. Using the model, created in Example 1 for prediction:

```
curl -H Accept:application/rdf+xml -X POST -d
dataset_uri=http://apps.ideaconsult.net:8180/ambit2/dataset/2?max=2
http://apps.ideaconsult.net:8180/ambit2/model/ToxTree%3A+Extended+Cramer+rules -iv

* About to connect() to apps.ideaconsult.net port 8180 (#0)
*   Trying 93.123.36.100... connected
* Connected to apps.ideaconsult.net (93.123.36.100) port 8180 (#0)
> POST /ambit2/model/ToxTree%3A+Extended+Cramer+rules HTTP/1.1
> User-Agent: curl/7.19.7 (amd64-portbld-freebsd8.0) libcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3
> Host: apps.ideaconsult.net:8180
> Accept:application/rdf+xml
> Content-Length: 67
```

```
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 303 See Other
< Server: Apache-Coyote/1.1
< Date: Sun, 14 Feb 2010 19:26:17 GMT
< Location: http://apps.ideaconsult.net:8180/ambit2/task/e8973705-174a-48e8-94b2-09c7f644beb3
< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
< Accept-Ranges: bytes
< Server: Restlet-Framework/2.0m6
< Content-Length: 0
<
* Connection #0 to host apps.ideaconsult.net left intact
* Closing connection #0
```

5.2.10 PLS

One way to understand Partial-least squares regression (PLS) is that it simultaneously projects the x and y variables onto the same subspace in such a way that there is a good relationship between the predictor and response data. Another way to see PLS is that it forms "new" x variables as linear combinations of the old ones, and subsequently uses these new linear combinations as predictors of y. Hence, as opposed to MLR PLS can handle correlated variables, which are noisy and possibly also incomplete.

General Information about the service
Service description The PLS web service enables the user to build regression models for a specific dataset.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/PLSregression
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2010 - February 09
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri=ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter. **numComponents** The number of components to compute. **performPrediction** Whether to update the class attribute with the predicted value. **preprocessing** Sets the type of preprocessing to use. **replaceMissing** Whether to replace missing values.
Response A task URI is provided if the service tries to calculate a regression model. The task URI can be queried (GET) for the resulting model URI. Otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
303	Redirect – the result can be found elsewhere.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/plsregression not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/PLSregression. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you posted are acceptable but some internal error has

	occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application.
Libraries used The service uses the WEKA software package.

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/23" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/12818" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/PLSregression
Example 2 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/158" -d "numComponents=2" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/12818" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/PLSregression

5.2.11 MaxTox (SL with JNU)

Published in 2006, Prakash and Ghosh elaborated the scope of the hypothesis that it may be possible to find a set of common scaffold(s) from a diverse compound set which contribute significantly (positively/negatively) towards biological activity. Our objective has been to extend this hypothesis to derive a predictive toxicity score, based on the MCS (Maximum Common Substructure) score with respect to clusters of compounds (based on toxicological endpoints).

The algorithm uses 2-D based QSAR to determine toxicity of molecules by comparing to a set of known toxic molecules. The QSAR in this case consists of finding descriptors from the database of toxic molecules using the maximum common substructure determination algorithm and then using these descriptors to develop a predictive model for toxicity. The test molecule is fed to this predictive model to get a score regarding its toxicity. At every level (mentioned below), the algorithm consists of

two parts – screening and rigorous graph matching. The main function of screening is to eliminate those molecules which are beyond some minimum similarity threshold (in terms of their graphs) so that the computationally complex graph matching is optimized. The algorithm consists of the following steps:

- Clustering of the molecules within this database based on toxicity endpoints (EP). Minimum Common Substructure (MCS) fragments are generated based on the well-known clique detection algorithm within each EP cluster.
- Comparing the query molecule to each cluster (EP based) and finding a fingerprint with respect to MCS fragments found in each cluster (from pair-wise comparisons within the molecules comprising the cluster)
- Using the fingerprint in a Machine Learning algorithm, to generate predictive models.
- Use the prediction model with the fingerprint of a new compound and predict a toxicity (classify) of the compound.

In accordance with our initial use case. currently the functionality provided is prediction of a single compound when posted to the model with a single parameter – the model number to use. In later iterations of the software, fine-grained control will be provided which is necessary for use cases like ToxPredict and leveraging other OpenTox APIs and data/structures.

General Information about the service
Service description The MaxTox service predicts a classification of a molecule (toxic / non-toxic – with relation to an EP) based on the fingerprint comprising of the occurrence/non-occurrence of MCS fragments found in pair-wise comparisons of molecules in an EP cluster.
URI http://opentox2.informatik.uni-freiburg.de:8080/MaxtoxTest/rest/model/{id} List: opentox2.informatik.uni-freiburg.de:8080/MaxtoxTest/rest/model
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Model
Date completed Initial version completed in Feb. 2010.
Partner responsible for the implementation SL-JNU
Contact within OT mr.surajit.ray@gmail.com, indirag@mail.jnu.ac.in, sunil@seascapelearning.com
Comments (including reviews)

Request/Response Information
Posted Parameters **compound_uri** (e.g. compound_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/compound/100846) which should be available in RDF format. This is a mandatory parameter.
Response A prediction task which when completed will provide a 0 (toxic) or 1(non-toxic) prediction for the compound.

Status Codes
200 Success – The request has succeeded and the prediction task has been generated.
400 Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the compound_uri.
404 The resource was not found.
500 Internal Server Error – The parameters you have posted are acceptable but some internal error has occurred.
502 Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503 The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application inside an Apache Tomcat server.
Libraries used CDK , R , Rserve , Jena , Restlet, MySQLConnector-J

Examples
Example 1 curl -X POST -d 'compound_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/compound/17611' http://opentox2.informatik.uni-freiburg.de:8080/MaxtoxTest/model/1

5.3 Clustering algorithms

5.3.1 K-means clustering

k-means clustering is an algorithm to classify or to group your objects based on attributes/features into K number of groups. K is a positive integer number. The grouping is achieved by minimizing the sum of squares of distances between data and the corresponding cluster centroids.

General Information about the service
Service description Simple k-means clustering
URI apps.ideaconsult.net:8180/ambit2/algorithm/SimpleKMeans
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed 2009/12/21
Partner responsible for the implementation IDEA
Contact within OT Jeliazkova.nina@gmail.com
Comments (including reviews)

Request/Response Information
Posted Parameters dataset_uri
Response Returns Task URL (/task/{id}) or Model URL (/model/{id}) in HTTP Location header.

Status Codes	
200	Success – The request has succeeded and a new model was generated. The URI of the model is returned in the HTTP Location: header.
202	Accepted – The request has been accepted, but model generation has not completed and a new model was generated. The URI of the task is returned in the HTTP Location: header.
303	Redirected (See Other) – The model generation is running. Returns Task URI in the HTTP Location: header.
404	Algorithm not found
500	Internal server error

Implementation Information	
HTTP Method	POST
Programming Language	Java ambit.svn.sourceforge.net/svnroot/ambit/trunk/ambit2-all/ambit2-www/src/main/java/ambit2/rest
Libraries used	Multiple (Ambit, Weka, Restlet among them)

Examples

Example 1 Creating clustering model. Note features in the dataset are explicitly set and the dataset_uri parameter value is URL encoded

```
curl -X POST -d
dataset_uri=http%3A%2F%2Fapps.ideaconsult.net%3A8180%2Fambit2%2Fdataset%2F7%3Ffeature_uris%5B%5D%3Dhttp%3A%2F%2Fapps.ideaconsult.net%3A8180%2Fambit2%2Ffeature%2F20185%26feature_uris%5B%5D%3Dhttp%3A%2F%2Fapps.ideaconsult.net%3A8180%2Fambit2%2Ffeature%2F20187%26feature_uris%5B%5D%3Dhttp%3A%2F%2Fapps.ideaconsult.net%3A8180%2Fambit2%2Ffeature%2F20186%26feature_uris%5B%5D%3Dhttp%3A%2F%2Fapps.ideaconsult.net%3A8180%2Fambit2%2Ffeature%2F20195
http://apps.ideaconsult.net:8180/ambit2/algorithm/SimpleKMeans -v

* About to connect() to apps.ideaconsult.net port 8180 (#0)
*   Trying 93.123.36.100... connected
* Connected to apps.ideaconsult.net (93.123.36.100) port 8180 (#0)
```

```
> POST /ambit2/algorithm/SimpleKMeans HTTP/1.1
> User-Agent: curl/7.19.7 (amd64-portbld-freebsd8.0) libcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3
> Host: apps.ideaconsult.net:8180
> Accept: */*
> Content-Length: 439
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 303 See Other
< Server: Apache-Coyote/1.1
< Date: Mon, 15 Feb 2010 06:51:55 GMT
< Location: http://apps.ideaconsult.net:8180/ambit2/task/75adf3aa-d17e-48e8-bfa3-295743916336
< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
< Accept-Ranges: bytes
< Server: Restlet-Framework/2.0m6
< Content-Length: 0
<
* Connection #0 to host apps.ideaconsult.net left intact
* Closing connection #0
```

Example 2 Retrieving the model by Task URL, received in Example 1

```
curl -X GET http://apps.ideaconsult.net:8180/ambit2/task/75adf3aa-d17e-48e8-bfa3-
295743916336 -v


* About to connect() to apps.ideaconsult.net port 8180 (#0)
*   Trying 93.123.36.100... connected
* Connected to apps.ideaconsult.net (93.123.36.100) port 8180 (#0)
> GET /ambit2/task/75adf3aa-d17e-48e8-bfa3-295743916336 HTTP/1.1
> User-Agent: curl/7.19.7 (amd64-portbld-freebsd8.0) libcurl/7.19.7 OpenSSL/0.9.8k zlib/1.2.3
> Host: apps.ideaconsult.net:8180
> Accept:application/rdf+xml
>
< HTTP/1.1 303 See Other
< Server: Apache-Coyote/1.1
< Date: Mon, 15 Feb 2010 06:53:18 GMT
< Location: http://apps.ideaconsult.net:8180/ambit2/model/9
< Accept-Ranges: bytes
< Server: Restlet-Framework/2.0m6
< Content-Length: 0
<
* Connection #0 to host apps.ideaconsult.net left intact
* Closing connection #0
```

5.4 Feature selection algorithms

5.4.1 Information Gain Attribute Evaluation

InfoGainAttributeEval evaluates the worth of an attribute by measuring the information gain with respect to the class.

InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute),

where H is the information entropy.

General Information about the service
Service description This service evaluates the worth of an attribute by measuring the information gain with respect to the class.
URI opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/InfoGainAttributeEval
OpenTox API Reference www.opentox.org/dev/apis/api-1.1/Algorithm
Date completed Initial version completed.
Partner responsible for the implementation Technische Universität München
Contact within OT kramer@in.tum.de
Comments (including reviews)

Request/Response Information
Posted Parameters **dataset_uri** (e.g. dataset_uri= ambit.uni-plovdiv.bg:8080/ambit2/dataset/39) which should be available in RDF format. This is a mandatory parameter. **binarizeNumericAttributes** Just binarize numeric attributes instead of properly discretizing them. **missingMerge** Distribute counts for missing values. Counts are distributed across other values in proportion to their frequency. Otherwise, missing is treated as a separate value. **numToSelect**

	Specify the number of attributes to retain. The default value (-1) indicates that all attributes are to be retained. Use either this option or a threshold to reduce the attribute set. **startSet** Specify a set of attributes to ignore. When generating the ranking, Ranker will not evaluate the attributes in this list. This is specified as a comma-separated list of attribute indexes starting at 1. It can include ranges e.g. 1,2,5-9,17. **threshold** Set threshold by which attributes can be discarded. Default value results in no attributes being discarded. Use either this option or numToSelect to reduce the attribute set.
Response	
	Once the features for a dataset are selected successfully, dataset URI is returned to the client with the selected features and the status is set to 200; otherwise an explanatory message is provided.

Status Codes	
200	Success – The request has succeeded and the requested features were generated. The URI of the dataset is returned within the response body.
400	Bad Request – Some parameter you provided is wrong or you did not post some mandatory parameter such as the dataset_uri.
404	The resource was not found – Check your spelling: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/infogainattributeeval is not identical to http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/InfoGainAttributEval. For a complete list of all available algorithms, check out http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/
500	Internal Server Error – The parameters you posted are acceptable but some internal error occurred.
502	Bad Gateway – The service was unsuccessful because while the server was acting as a client, it received an unsuccessful response. In such a case, it seems that some other server is down.
503	The service is not available for the time being – Try again later!

Implementation Information
HTTP Method POST
Programming Language The project was built in Java and runs as a standalone application.
Libraries used The service uses the class InfoGainAttributeEval which is included in WEKA (version 3.6.0).

Examples
Example 1 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/158" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/29634" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/InfoGainAttributeEval
Example 2 curl -X POST -d "dataset_uri=http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/23" -d "prediction_feature=http://ambit.uni-plovdiv.bg:8080/ambit2/feature/12818" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/InfoGainAttributeEval

6. Conclusions

This document summarizes the work that has been accomplished within the OpenTox Framework regarding the development of the initial prototype of QSAR algorithms. A key decision towards this implementation was the adoption of the REST architectural style, because it is suitable for achieving three important goals: independent deployment of components, ease of standardised communication between components and generality of interfaces. These advantages will enable the development and integration of additional algorithms in the future, which may be offered not only by OpenTox partners but by third-party developers as well. Ongoing maintenance and addition of novel predictive algorithms relevant to predictive toxicology will contribute to the long-term sustainability of OpenTox in generating valuable resources for the user scientific community.

A significant amount of time was invested by the developers in the design of the OpenTox Framework to create an extensible, interoperable, well-engineered computing platform for predictive toxicology, which is currently absent but sorely needed to address current and future industry and regulatory needs, and to advance the safety assessment of products to ensure human health. The project is ahead of schedule regarding individual implementation of algorithms, since more algorithms than initially planned have already been prototyped. In particular, many descriptor calculation algorithms and QSAR modeling methods have already been implemented and incorporated within OpenTox. These include methods provided by OpenTox partners and algorithms contained in other state-of-the-art projects such as WEKA and CDK. Descriptor calculation algorithms are able to generate both physico-chemical and sub-structural descriptors. QSAR modeling methods cover a wide range of approaches and address many user model building requirements, since they include regression and classification algorithms, eager and lazy approaches, and algorithms producing more easily interpretable and understandable models. The initial prototype also includes implementations of clustering algorithms and feature selection tools.

Continuing effort will be carried out by all OpenTox partners to meet current academic and industry challenges regarding interoperability of software components and integration of algorithm and model services within the context of tested use cases. The experience we have gained during this work will help speed up the development process towards this direction. The approach to interoperability and standards lays a solid foundation to extend application development within the broader developer community to establish computing capabilities that are sorely missing in the field of predictive toxicology today, and which are holding back advances in both R&D and the application of R&D project outcomes to meet industry and regulatory needs.

7. Appendix. Algorithm Type Ontology