Predictive Model Representation and Comparison: Towards Data and Predictive Models Governance

Mokhairi Makhtar, Daniel C. Neagu, Mick Ridley
School of Computing, Informatics and Media, University of Bradford

Abstract— **The increasing variety of data mining tools offers a large palette of types and representation formats for predictive models. Managing the models becomes then a big challenge, as well as reusing the models and keeping the consistency of model and data repositories because of the lack of an agreed representation across the models. The flexibility of XML representation makes it easier to provide solutions for Data and Model Governance (DMG) and support data and model exchange. We choose Predictive Toxicology as an application field to demonstrate our approach to represent predictive models linked to data for DMG. We propose an original structure: Predictive Toxicology Markup Language (PTML) offers a representation scheme for predictive toxicology data and models generated by data mining tools. We also show how this representation offers possibilities to compare models by similarity using our Distance Models Comparison technique. This work is ongoing and first encouraging results for calculating PTML distance are reported hereby.**

I. INTRODUCTION

The processes of generating predictive models involve data preparation, checking of data quality, reduction, modelling, prediction, and analysis of results. Generating high-quality predictive models is a time consuming activity because of the tuning process in finding optimum model parameters. Each benchmark model is trained to find which attributes and model parameters are the best in producing a better model. The tuning process involves the selection of optimum model parameters such as the number of fold-cross validation and the classifier type. Selection of the optimum attributes from the data set is another step of the tuning process. This will be iterated until the best combination of parameters is selected to generate the best or a better model to be added into the erpository.

In the case of dataset updates, predictive models related to the older dataset become unreliable. This evolution of training datasets always happen in application domains such as banking, where transactions are updated regularly, and also in toxicology, where experiment circumstances and new compounds are added. To generate new and reliable predictive models for the updated dataset, the iteration process of tuning and finding the best combination of attributes and model parameters must take these changes into account. This makes it necessary to recall the predictive model generation step for an up-to-date model repository iteration.

This continuous process shows that there may be thousands of data mining models related to a single dataset shared among data mining researchers, generating versions of predictive models and related datasets. Thus, to monitor and maintain changes between data and models becomes even more challenging.

There is a need to define the relationship between data and models, so that the iteration process of generating new predictive models integrates consistently in the modelling framework and this evolution needs also be recorded. These repositories of data mining models should keep information on historical developments which are also valuable for analysis.

Another challenge here is how to share those models between researchers. Existing models are represented in various platforms, for example text files, relational database or different internal format produced from data mining tools (e.g. `.arff` produced by Weka and `.fis` produced by Matlab). XML is a key to the answer where the models can be published through the web in standard form and can be accessed easily later. For that reason we propose our representations in XML as a bridge and a flexible solution to deal with the current diversity of model representations available in the literature.

The other challenge that arises here is whether available models can be analysed and interpreted so that information they store can be used later to generate better predictive models. Since there is information stored in the previous models available in repositories, there should be provided the possibility of selecting the best or most suitable models from the collection of models based on individual requirements and needs. This can be done in many ways such as searching the model with a selection of criteria, comparing the performance of existing models or opening competitions between models. These approaches have often been proved to achieve better predictive performance compared to producing a single predictive data mining model [1].

In this paper we propose a flexible data and model representation in a more general framework towards data and model governance. In section two we introduce the concept of DMG and in section three we describe the framework and structures of PTML representation. The automation of generated predictive models is described in section four. Section five introduces a first DMG application: a comparison technique proposed to calculate distances between models. Some encouraging results of the technique can be found in section six. Conclusions and further work are provided in the last section.

II. DATA AND MODEL GOVERNANCE

A global view of predictive modelling must involve data and models. Thus this valuable combination of data and models needs proper management. Data Governance is defined by IBM as the quality control discipline for assessing, managing, using, improving, monitoring, maintaining, and protecting organizational information [2]. The process for generating predictive models involves steps of Data Preparation, Data Reduction, Data Modelling and Prediction, Evaluating and Validating the Model, and Implementing and Maintaining the Model. The processes of predictive modelling need to be properly managed and controlled because of the consequences in the decision making. This is why we propose progressing towards Data and (predictive) Model Governance (DMG). We define DMG as the set of quality control processes for assessing, managing, using, improving, monitoring, maintaining, and protecting data and (predictive) model information.

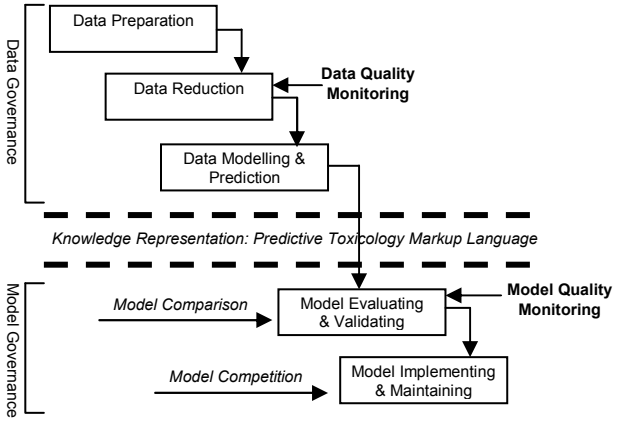

Fig. 1. Data and Model Governance framework

In Fig. 1 we depict the whole task of predictive modelling towards Data and Model Governance framework from the beginning process of generating predictive models until implementing the models. This iteration process should be governed in every stage in order to maintain the quality of predictive models generated for which we propose this framework. The models generated are represented in PTML for further processing. In this paper we show how PTML can be applied for DMG using our methodology.

III. MODEL REPRESENTATION

Results generated from the data mining can be represented in many forms as predictive models. The purpose of our integrative approach for data and model representation is to visualize the model, extract the parameters of the models, process and manage the models in relation to the available data. More significant processing can be further done to the models, such as comparison, selection and competitions between them to respond to subsequent tasks. We outline below a number of languages that represent predictive models.

PMML (Predictive Model Markup Language) [3, 4] is a standard XML-based language to represent predictive models which allows models sharing between applications. It was established by Data Mining Group and has 4 components: Data Dictionary, Mining Model, Transformation Dictionary and Model Statistics. PMQL (Predictive Modelling Query Language) is a specialized query language for interacting with PMML documents. It is embedded within DeVisa framework, which provides functions such as scoring, model comparison, model composition, searching, statistics and administration through a web service interface [5]. Augustus is another PMML-compliant scoring engine [6] developed in Python, which extended the PMML standard components.

The Hybrid Intelligent Systems Markup Language is a XML proposal for knowledge representation, data exchange and experimental data analysis, based on a modular implicit and explicit knowledge-based intelligent system [7, 8].

PToxML (Predictive Toxicology Markup Language) is used to describe chemical information related to predictive toxicology [7, 8]. It has three main sections: Header section, Chemical Identification section and Data section.

ToxML [9] is an XML database standard based on toxicity controlled vocabulary for use in database standardization. It was developed by scientists at Leadscope Inc. for applications in areas such as genetic toxicity, carcinogenicity, chronic toxicity etc.

A. Predictive Toxicology Markup Language (PTML)

We propose Predictive Toxicology Markup Language based on the model proposed in [7] to provide solutions in data and model representation for toxicology data. PTML represents data mining models in a standard format and can be simply manipulated for search and comparison. It also describes predictive toxicology data and the associated models generated by data mining processes.

We use Weka and Java as the two main tools in generating predictive models process and converting them into PTML. Currently PTML is an open structure and the tags are still under development: a *generateWekaModel* function is used to retrieve outputs from Weka and *PTMLTranslation* is used to translate the output from Weka to PTML.

B. Model Structures

PTML was proposed to make predictive models easier to analyse and interpret. The PTML structure currently consists of six elements: Model Description, Model Attributes, Model Parameter, Model Performance, Class Statistic Performance and Confusion Matrix (see Table 1). We chose the Confusion Matrix as the most comprehensive source of performance information for further processing.

Model Description

This section describes the general information of the predictive model with attributes like date when the model was generated, descriptions of model and file name for Weka model type (see Table 2).

Model Attributes

This section describes the data set and attributes used for the generation of the predictive models. The information related includes a file name of the data set the model is generated from, the number of instances and the number of attributes (see Table 3). We emphasize the inclusion of the data source into the model representation, thus further operations to compare models by the source data could be defined for model comparison. This section is different from the PMML representation: information related to dataset preprocessing such as Feature Selection Algorithms and Searching Method of Feature Selection is clearly represented in PTML as it will affect the performance and quality of the generated predictive models.

Model Parameter

Another important part of a predictive model is the parameter settings. Information like classifier type, number of folds and seed used to distinguish between models. This information is useful for describing or regenerating predictive models (see Table 4).

Model Performance

The section of model quality is a wrapper around various elements and holds related results generated from a specific dataset, although it is possible to regenerate the model. Statistical performances for the model generated are shown in this section, such as correctly classified instances, mean absolute error and root mean squared error, and conclusions can be made about the model's quality (see Table 5).

Class Statistic Performance

Further performance for each class attribute is stated in this section such as the true positive rate, false positive rate, precision, recall and receiver operating characteristic (ROC) area (see Table 6).

Confusion Matrix

This is the most important element in generating classification models. It can give an overview of correct and incorrect classifications to the class attribute (see Table 7).

TABLE 1
THE PTML DOCUMENT STRUCTURE

```
<dataMiningModel>
        <modelDescription>
        :
        </modelDescription>
        <modelAttributes>
        :
        </modelAttributes>
        <modelParameter>
        :
        </modelParameter>
        <modelPerformance>
        :
        </modelPerformance>
        <classAttribute>
        :
        </classAttribute>
        <ConfusionMatrix>
        :
        </ConfusionMatrix>
</dataMiningModel>
```

TABLE 2
THE MAIN SECTION OF PTML DOCUMENTS

```
<modelDescription>
        <Name>DM</Name>
        <Date>25-12-2008</Date>
        <Version>Ver1.1</Version>
        <Author>Mokhairi</Author>
        <Description>Testing Autogenerated
        Model From Weka
        </Description>
        <wekaModel>wekaModel20.model
        </wekaModel>
</modelDescription>
```

TABLE 3
MODEL DATA AND ATTRIBUTES INFORMATION

```
<modelAttributes>
        <DataSet>APC_Recon-(C)Mallard_Duck-
        Raw_Data-Training.arff</DataSet>
        <FeatureSelectionAlgorithm>
        CfsSubsetEval
        </FeatureSelectionAlgorithm>
        <FeatureSearchMethod>BestFirst
        <FeatureSearchMethod>
        <TotalNumberInstances>24.0
        </TotalNumberInstances>
        <NumberOfAttributes>184
        </NumberOfAttributes>
        <NumberOfAttributesSelected>7
        </NumberOfAttributesSelected>
        <Features>
                <Name>Del(Rho)NA5</Name>
                <Type>Numeric</Type>
        </Features>
        :
</modelAttributes>
```

TABLE 4
MODEL PARAMETER SETTINGS

```
<modelParameter>
        <Option Classifier=
        "weka.classifiers.trees.J48">
        </Option>
        <Option TrainingType>
        10fold-cross-validation</option>
        <Option Fold="10"></Option>
        <Option Seed="1"></Option>
</modelParameter>
```

TABLE 5
OVERALL MODEL STATISTICAL PERFORMANCE

```
<modelPerformance>
        <modelType>Classification <modelType>
        <CorrectlyClassifiedInstances>20.0
        </CorrectlyClassifiedInstances>
        <IncorrectlyClassifiedInstances>4.0
        </IncorrectlyClassifiedInstances>
        <Accuracy>83.33</Accuracy>
        <Kappa>0.71</Kappa>
        <MeanAbsoluteError>0.15
        </MeanAbsoluteError>
        <RootMeanSquaredError>0.33
        </RootMeanSquaredError>
        <RelativeAbsoluteError>40.77
        </RelativeAbsoluteError>
        <RootRelativeSquaredError>76.59
        </RootRelativeSquaredError>
</modelPerformance>
```

TABLE 6
CLASS STATISTICAL PERFORMANCE

```
<classAttribute>
       <Name>contact-lenses</Name>
       <Class>soft</Class>
       <Details>
              <TPRate>1.0</TPRate>
              <FPRate>0.053</FPRate>
              <Precision>0.833</Precision>
              <Recall>1.0</Recall>
              <FMeasure>0.909</FMeasure>
              <ROCArea>0.947</ROCArea>
       </Details>
       <Class>hard</Class>
       :
       :
</classAttribute>
```

TABLE 7
CONFUSION MATRIX TABLE

```
<ConfusionMatrix>
       <Array>Class1   Class2  </Array>
       <Array>5   0    Class1</Array>
       <Array>0   3    Class2</Array>
</ConfusionMatrix>
```

IV. GENERATED PREDICTIVE MODELS

The automatic generation of predictive models is well addressed in the literature mainly in work related to Hybrid Intelligent Systems [7]. One of the main motivations is tuning generated models, and adapting them to further problems is not an easy task. Rich et al. [1] addressed models generation for use in ensembles of models. They generated diverse sets of models by using seven different algorithms. The algorithms used were Support Vector Machines (SVMs), Artificial Neural Nets (ANNs), Memory based Learning algorithms: k-NN, Decision Trees (DT), Bagged Decision Trees (BAG-DT), Boosted Decision Trees (BST-DT) and Boosted Stumps (BST-STMP). All algorithms were using different parameter settings. About 2000 models were trained and applied to test sets.

For this research, we generated collections of models using algorithms implemented in Weka [10], such as k-nearest kneighbours classifier (weka.classifiers.lazy.IBk), decision trees (weka.classifiers.trees.J48) and numerical prediction (weka.classifiers.rules.JRip) The predictive models were applied to various data sets from the EC FP5 project Demetra [11], CSL [12] and TETRATOX [13]. We generated 72 predictive models with different combinations of datasets, algorithms, and model parameters. A consistent approach of model and data repository with a formal representation of all these objects has not been addressed to date, as far as we are aware. We propose the following steps in order to generate, represent and store data and associated models for further systematic use.

A. XML-based Predictive Models

We propose that all predictive models, in XML format, are represented automatically. These are the steps taken to automate the generation of XML based language for predictive models:

a. Prepare Data

The important step in the data mining process to generate a predictive model is data preparation. The amount of data is normally checked for mistakes, out of range values or impossible data combinations. Results can be misleading if data are not properly prepared. Other important processes of pre-processing of dataset are Feature Selection and Searching Method of the Feature Selection. These processes will make sure the attributes to be selected for the predictive models are highly correlated with the output.

b. Choose Model and Parameter Settings

Different combinations of input settings will be used to generate predictive models. The settings such as classifier type and number of folds may affect the performance of the generated models.

c. Generate the Model and Test its Performance

In this case, Weka has been used to generate the data mining models but many other model generation tools may be also used. From the input given (data and model parameters), models are generated automatically and tested against test sets. These models are stored in text files of internal format (e.g. Weka Generated Model with file extension `.model`).

d. Generate XML Model

The internal storage representation has to be converted into the XML format for later processing and analysis. We propose PTML to bridge various model formats.

e. Test the model representation

Models in XML format will be tested by retrieving the model using XQuery to check that there are no syntactical errors in their representation.

f. Publish the models

The XML model can be published and stored in the repository for further processing tasks. In our case, the PTML model is used for data and model representation in repositories.

V. XML MODEL COMPARISON

The most suitable model(s) from the collection of models can be selected by searching the entire range of models based on input given. Gorea [2] compared two models through exact references (Model ID) and comparison type (syntactic or semantic approach). The syntactic comparison is where two PMML models are compared through a XML differencing approach. The semantic comparison involves a schema compatibility check, a function comparison or a performance comparison using different metrics as described below.

We compare models by calculating distances between models and comparing the information gathered from PTML. Based on PTML format, we define the Predictive Model $M = \{I,F,O\}$ consisting of Input I, Function F and Output O, where I is the input used to generate the predictive models and all the information related to certain dataset. All datasets for this paper use the Weka format (.arff). From PTML, `<modelAttributes>` (Table 3) is the main information of input for the predictive models. It consists of information such as:

<FeatureSelectionAlgorithm>, <FeatureSearchMethod> and <Features>:
$I = \{i1, i2, \{i3\}\}$ where
i1 is the name of <FeatureSelectionAlgorithm>, *i2* is the name of <FeatureSearchMethod> and *i3* is the list of names of selected features <Feature> used for the generated model.

F is a function referring to information from <modelParameter> (Table 4) such as Classifier and TrainingType:
$F = \{f1, f2, f3\}$
where *f1* is the classifier name, *f2* is the TrainingType category and *f3* is the category of <modelType> (e.g. classification or regression).

O is the tuple of results from the generated predictive model and refers to the information from <modelPerformance> (Table 5) and <classAttribute> (Table 6). The <modelPerformance> is the overall performance of a predictive model and <classAttribute> is the performance for each class of a classification model:
$O = \{o1, o2\}$
where *o1* is the value of <Accuracy> for a predictive model, *o2* is the number of class attributes.

The comparison between models requires measuring the similarity between them. From the definition $M = \{I,F,O\}$, the values of *I* and *F* are nominal and those of *O* are numerical. We applied the Hamming distance to calculate the distance for *I* and *F* because of the nominal values and Euclidean distance for the output *O*.

The Hamming distance between two objects measures the disagreement between two vectors $h_{ij} = q + r$ where *q* is the number of variables with value "1" for the *i*-th object and "0" for the *j*-th object, and *r* is the number of variables with value "0" for the *i*-th object and "1" for the *j*-th object.

The Euclidean distance d_{ij} is commonly used to find a distance between two objects with quantitative values. It calculates the root of squared differences between coordinates of a pair of objects: $d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$. It also can be used to find distances between two predictive models. For this paper, the distances for the predictive models were calculated for the classification models as described below.

First, we find all the similar values of I_{Mi}, I_{Mj}, and F_{Mi}, F_{Mj}. We compare every pair of the variables. If the pair value of the variables is similar, then we set the vector value $CV_{Ii:j}$ to "0" and if it is different we assign a "1" to the vector. This follows the Hamming distance rule that calculates the disagreement between two vectors.

Given the values of models *i, j*:

I_{Mi}={"CfsSubsetEval","BestFirst",{"abc1","xyz","xyz1"}};
I_{Mj} = {"CfsSubsetEval", "GreedyStepwise",{"abc1"}};

F_{Mi} = {"DecisonTree", "10-Folds, "classification"}
F_{Mj} = {"NeuralNet", "10-Folds","classification"}

O_{Mi} = {0.70, 4}
O_{Mj} = {0.91, 4}

the results of comparison are saved in the vectors:

$CV = [CV_I; CV_F]$
$CV_{Ii:j} = [0,1,\{0,1,1\}]$
$CV_{Fi:j} = [1,0,0]$

To compute the distance of *I* and *F,* add all the ones and zeros in the vectors as follows:

$hI_{ij} = 0+1+0+1+1 = 3$
$hF_{ij} = 1+0+0 = 1$

The distances of *I* and *F* for models M_{ij} are 3 and 1. The distance for *O,* based on Euclidean distance, is:

$$0_{ij} = \sqrt{\sum_{k-1}^{n}(x_{Mik} - x_{Mjk})^2} = \sqrt{(O_{Mi} - O_{Mj})^2}$$

Finally we propose Distance Models Comparison d_{Mij} to find distances between models:

$$d_{Mij} = hI_{ij} + hF_{ij} + O_{ij}$$

The result in this case shows that the distance between two models M_i and M_j is 3 + 1 + 0.21 = 4.21. The higher value of d_{Mij} will be considered as the bigger distance between the two models. If $d_{Mij} = 0$, it means that models are identical.

VI. Results and Discussion

In this section we explore the results saved using the PTML structure. We have 72 models from our collections and 24 of the models are related to the input. The models were generated from a group of predictive toxicology data sets [7] whereby each group of data set had been run through data preparation and reductions processes. The original data sets had been split into 70% (training data set) and 30% (testing data set).

We also examine the relationship of data sets that had a feature selection. Feature selection algorithms applied to the datasets were Correlation-based Feature Selection (CFS), Classifier Sebset Evaluator(Clsfr) and Consistency Subset Evaluator(Cnstcy). The searching methods for every feature selection were Best First Search (Best), Genetic Search (Gen) or Greedy Stepwise (Greedy).

We used feature selection to find sets of attributes that are highly correlated with the target classes. These data sets had also been used by Trundle in his dissertation [14]. Each data set was run using Weka with 10-fold cross validation from which we chose the following classifiers: weka.classifiers.lazy.IBk, weka.classifiers.trees.J48, weka.classifiers.rules.Jrip.

The results highlight several interesting trends across both the datasets and the range of modelling algorithms. The results are shown Table 8 and Table 9. We also found that the performance of models in terms of accuracy using train and test datasets is consistently lower than performance from the 10-fold cross-validation approach. Another important finding regards the accuracy results: using feature selection algorithms we obtained higher accuracy than using the original features set. This trend is also shown in Table 8 and Table 9.

This experiment shows significant results in comparing the relevant models from the database of models and selecting the best model based on highest accuracy. From the results, J48 is the outstanding classifier which is 46.1% train using raw dataset and BayesNet(Bnet) is 53.8% is the best classifier for datasets with feature selection. J48 shows a good average for all categories of datasets.

TABLE 8
MODEL ACCURACIES ON DATASETS WITH NO FEATURE SELECTION AND TEST OPTIONS 10-FOLD CROSS-VALIDATION

DataSet	*J48*	*IBK*	*BNet*	*JRip*	Average
Mallard Duck	46.7	40	43.3	36.7	41.7
Bee	44.7	43.8	45.7	41	43.8
Daphnia	47	48.4	45.4	45.8	46.7
Average	46.1	44.1	31.5	41.2	44.1

TABLE 9
MODEL ACCURACIES ON DATASETS WITH FEATURE SELECTION AND TEST OPTIONS 10-FOLD CROSS-VALIDATION

DataSet	*J48*	*IBK*	*BNet*	*JRip*	Average
Mallard Duck	43.3	38.3	58.3	38.3	44.6
Bee	47.6	49.5	52.4	47.6	49.3
Daphnia	52.6	49.6	50.7	50.7	50.9
Average	47.8	45.8	53.8	45.5	48.3

TABLE 10
MODEL DISTANCE MATRIX OF THE 7 MODELS OBTAINED FOR THE DAPHNIA DATA SET

ID	*M1*	*M2*	*M3*	*M4*	*M5*	*M6*	*M7*
M1	0	22.2	19.2	28.1	19.2	19.2	28.1
M2	22.2	0	11	22.1	12	13	24.1
M3	19.2	11	0	25.1	1	2	27.1
M4	28.1	22.1	25.1	0	24.1	25.1	3
M5	19.2	12	1	24.1	0	1	27.1
M6	19.2	13	2	25.1	1	0	26.1
M7	28.1	24.1	27.1	3	27.1	26.1	0

Table 10 shows the result of the Models Comparison using Distance Models Comparison technique that we proposed. The values are distances between all possible pairs of the 7 models. Note that models M3 and M5 are very close to each other. A distance value of "1" means that there is only a value different between the two models. This indicates that the two models are quite similar because of the high level of similarity. Models M4 and M7 are quite close to each other but they are not similar to model M1 because the distance is bigger.

VII. CONCLUSIONS

Extraction of data mining models is an important issue. Producing the most useful data mining models is incomplete without interpreting and processing knowledge from the models. Extracting the model parameters from XML representation illustrates that the models are valuable in terms of understanding, and manageable for further usage. PTML representation gives an useful solution for extracting knowledge form data mining. The representation also offers possibilities to compare the similarity of models e.g. using Distance Models Comparison. Further work in understanding and analysing model performance metrics of data mining models will give more advantages in selecting the best model from the collections of models, from a fitness for purpose perspective.

REFERENCES

[1] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble Selection from Libraries of Models," in *2004 Procs of the 21st International Conference on Machine Learning*, pp. 137-144.

[2] IBM. Available: http://www-01.ibm.com/ software/tivoli/ governance/servicemanagement/ data-governance.html."

[3] "Data Mining Group. Available: http://www.dmg.org/rfc/."

[4] D. M. Group, "PMML 3.2 - Model Explanation Documents," 2008.

[5] D. Gorea, "Dynamically Integrating Knowledge in Applications. AnOnline Scoring Engine Architecture," *Advances in Electrical and Computer Engineering*, vol. 8(15), pp. 44-49, 2008.

[6] J. Chaves, C. Curry, R. L. Grossman, D. Locke, and S. Vejcik, "Augustus: The Design and Architecture of a PMML-Based Scoring Engine," in *2006 Procs of the 4th Int'l Workshop on Data mining standards, services and platforms DMSSP'06. New York: ACM*, pp. 38–46.

[7] D. Neagu, M. Craciun, Q. Chaudhry, and N. Price, "Chapter 10. Knowledge Representation for Versatile Hybrid Intelligent Processing Applied in Predictive Toxicology," *Life Science Data Mining*, pp. 213-238, 2007.

[8] D. Neagu, M. V. Craciun, S. A. Stroia, and S. Bumbaru, "Hybrid Intelligent Systems for Predictive Toxicology - a Distributed Approach," in *2005 Procs of the 5th International Conference on Intelligent Systems Design and Applications (ISDA '05)*, pp. 26 - 31.

[9] "Leadscope Inc Available: http://www.leadscope.com/toxml/."

[10] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," in 1999 *Procs of the ICONIP/ANZIIS/ANNES'99 Workshop Emerging Knowledge Eng. and Connectionist-Based Information Systems*, pp. 192-196.

[11] "DEMETRA Project Available: http://www.demetra-tox.net/."

[12] "CSL, Central Science Laboratory Available: http://www.csl.gov.uk/"

[13] "TETRATOX, Available: http://www.vet.utk.edu/TETRATOX/index.php. 2008."

[14] P. Trundle, "Hybrid Intelligent Systems Applied to Predict Pesticides Toxicity - a Data Integration Approach" PhD Thesis. School of Informatics, University of Bradford, UK., 2008.