Pattern Recognition Engineering

Faisal Shafait[1], Matthias Reif[1], Christian Kofler[1], and Thomas M. Breuel[2]
[1]Multimedia Analysis and Data Mining Competence Center
German Research Center for Artificial Intelligence (DFKI GmbH)
D-67663 Kaiserslautern, Germany
{faisal.shafait,matthias.reif,christian.kofler}@dfki.de
[2]Image Understanding and Pattern Recognition (IUPR) Research Group
Technical University of Kaiserslautern, 67663 Kaiserslautern, Germany
tmb@informatik.uni-kl.de

***Abstract*— This paper outlines some key software components developed in the Pattern Recognition Engineering (PaREn) project. The goal of the PaREn project was to create the methods and tools necessary allowing non-experts to use, train, test, and deploy pattern recognition and machine learning modules in real-world software systems. A major effort in the PaREn project was therefore automating parameter optimization, model selection, machine learning system construction, and supporting rapid testing, validation, and on-line adaptivity. To deliver our technologies as open source, we chose RapidMiner as the software platform. Therefore, major software components developed in PaREn are provided as RapidMiner extensions. The expected benefits are a far wider usage of pattern recognition and machine learning methods, leading to both better quality of the decisions and behaviors of software systems, as well as lower development costs.**

I. Introduction

Although many new pattern recognition and machine learning methods have been developed over the last two decades, integrating those methods in real-world software systems is still a trying experience for both users and software developers. We believe that this gap is due to a lack of tools and methods supporting such integration, as well as to the high level of expertise demanded from software developers in areas such as model construction, model selection, and validation.

Perhaps the biggest obstacle to the adoption and integration of pattern recognition and machine learning methods into real-world software systems is the mathematical complexity and sophistication required for adapting them to particular problems. This is not primarily a software engineering issue, it is a fundamental problem with the methods themselves: they usually have many parameters and their behavior is highly sensitive to how pattern recognition modules are interconnected. Furthermore, while many existing software components also may have many parameters that affect their behavior, those parameters generally correspond to concepts that are meaningful to application developers and can be optimized via trial and error. The goal of the PaREn project is to develop technologies for automating and supporting model construction and selection in real-world settings. Our vision is that this will take pattern recognition methods out of the hands of academics and specialists and make them far more accessible to real-world software developers.

Each of the major stakeholders in the development of pattern recognition systems–end users, application engineers, software developers, pattern recognition researchers, and students–should be able to focus on the relevant aspects of development and deployment of such systems without having to acquire significant expertise outside their own areas of concern. We can illustrate this through a number of use cases and scenarios.

Consider an **application engineer** who would like to develop a pattern recognition system with data at hand but does not have enough expertise to choose and configure an appropriate pattern recognition software pipeline. A pipeline in this context may e.g. consist of steps for providing data, preprocessing and classification. The application engineer uploads a labeled dataset to the PaREn tools. After analysis of the data, the most appropriate Pattern Recognition (PR) Pipeline is chosen from a set of candidates in a Case Base. The application engineer may confirm or correct the selection of the pipeline. The selected pipeline can then be optimized according to the uploaded data.

In addition to the functionality for standard users, an **expert user** needs to have more control over the involved steps and actual implementations of pattern recognition pipelines. An expert user can review pipelines in depth and replace or modify the individual algorithms involved. Modified versions of an algorithm or new algorithms can be committed and automatically evaluated. Using the experiment repository, the performance of the respective algorithm can be compared to previous revisions.

II. Related Work

The pattern recognition community has recognized the importance of pattern recognition *engineering* on various occasions before, which we would like to illustrate with two quotes:

"It is worth noting that Theo Pavlidis, in his King-Sun Fu lecture during ICPR 2000, recommended PR researchers to use the engineering paradigm in their work, because their final goal is the design and construction of PR machines." [1], [2] "According to computer science encyclopedia and some famous textbooks, pattern recognition (PR) can be defined as

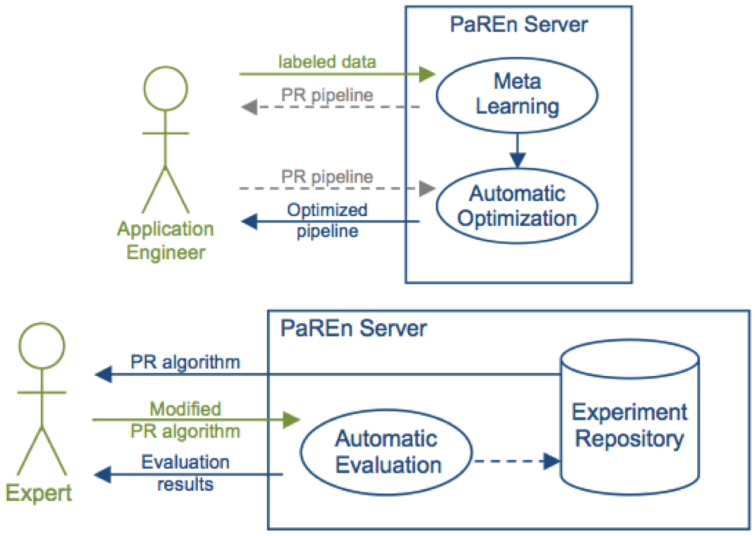


Fig. 1. The main use cases of the PaREn tools for Application Engineers and Experts.

the discipline that studies theories and methods for designing machines that are able to recognize patterns in noisy data [...]. Although this is only one of the possible definitions for PR, it points out well the 'engineering' nature of this scientific discipline, because it says that the final goal of PR is the design of 'machines'."[1]

Most relevant work to PaREn has been carried out in pattern recognition on model selection and model validation. Of particular note is the EU STATLOG project [3], [4], *comparing algorithms* from symbolic learning, statistics, and neural networks on twelve data sets. The project focused on classification problems and performed an extensive evaluation and comparison of different methods on common datasets. The STATLOG project did not simply report the relative performance of different methods on different datasets, but also applied statistical techniques to the performance of the different algorithms and datasets themselves. This was used, for example, to automatically generate a taxonomy of pattern recognition methods based on their performance on different datasets, and of datasets based on how close the performance of different methods matched on those datasets.

STATLOG also included some work on *meta-level learning* and model selection and created an "Application Assistant", a rule-based expert system making recommendations to users as to which pattern recognition methods to use, combining both empirical and *a priori* information. The Application Assistant was considered "promising" by its developers and has been able to make useful recommendations. However, the Application Assistant does not deal with the construction of entire pattern recognition systems (preprocessing, classification, post-processing), it only makes recommendations. Furthermore, the Application Assistant does not appear to have been integrated into the development process itself; that is, it would make recommendations, but it was up to the developer to put those recommendations into practice.

Within the EU project MetaL [5], an open source software for calculating measures of datasets was developed. This data characterization toolkit (DCT) calculates several features, that describe the dataset itself. These meta-features contain skewness, kurtosis, SD ratio, and class entropy of the data and can be used to predict the applicability of pattern recognition algorithms. Additionally, landmarking was used for characterizing datasets [6]. Landmarking utilizes simple, fast computable classification algorithms and uses their achieved accuracy values as features of the dataset.

Recently, developments in the areas of *automatic algorithm selection*, model selection, and parameter optimization have been tested as part of the Model Selection Workshop (2006) [7]. The conclusions of the workshop are summarized in [8]. Generally, simple cross-validation techniques were found to work fairly reliably and predictably. The authors and organizers concluded that "there is still a gap to be filled between theory and practice in this game of performance prediction and model selection". Generally, work in recent years has attempted to explore a large number of different algorithms and approaches to meta-learning, but little more work appears to have been delivering these new methods to real-world environments.

III. Choosing a Software Platform

We performed a thorough review and evaluation of existing open source machine learning tools to incorporate our methods into existing tools as much as possible. Twenty-two requirements were defined and weighted according to their importance for PaREn.

Table I shows an excerpt of the evaluation results and the final overall rating. RapidMiner and KNIME fulfilled the most important requirements. The final decision has been supported by the fact that RapidMiner has evolved as the de facto standard in open source data mining tools. RapidMiner was the only open source tool in the top five

TABLE I

EXCERPT OF THE EVALUATION OF PATTERN RECOGNITION AND DATA MINING TOOLS. EACH SOFTWARE HAS BEEN RATED BETWEEN 1 (BAD) AND 5 (VERY GOOD) FOR EACH OF THE WEIGHTED REQUIREMENTS.

Requirement	**RapidMiner**	**Orange**	**KNIME**	**KEEL**	**Weka**
reliable implementation of the major algorithms for classification and prediction	5	3	5	5	5
consistent implementation of a persistent pipeline concept	5	4	4	2	3
programmatic access to all relevant components and parameters (API)	5	5	5	1	5
extensibility with plug-ins	5	5	5	2	3
software runs stable and is robust against errors	5	4	5	4	4
implementation and integration of data preprocessing algorithms	5	3	5	5	5
API is well-documented	5	4	5	1	5
good visualization of data and results	5	3	4	2	3
implementation and integration of clustering algorithms	5	3	4	2	4
handling of multiple data formats and sources	5	3	3	5	5
support for PMML (Predictive Model Markup Language)	3	1	5	1	4
high acceptance in the community	5	4	4	4	5
command line interface	5	3	5	4	3
intuitive HCI	4	3	5	2	3
support for multiple platforms and operating systems	5	3	4	4	5
...	...	...	...	...	...
Overall rating	**4.83**	3.52	4.00	2.75	3.52

in the KDnuggets Poll in 2009 [9]. KDnuggets is highly accepted in the data mining community and creates yearly surveys of the most popular software tools. An interesting aspect of RapidMiner is its support of the PMML (Predictive Model Markup Language) standard. With PMML, it is straightforward to develop a model on one system using one application and deploy the model on another system using another application. Therefore, the results of experiments performed in RapidMiner can be directly used in other PMML compliant systems.

Based on the evaluation and the high acceptance in the community, we decided to use RapidMiner as basis and integration framework for the PaREn software results. Most of the software components presented here build on, extend or integrate RapidMiner software.

IV. SOFTWARE COMPONENTS

A. Comparison of Classifiers over UCI datasets

For the comparison and evaluation of classifiers we created a so called case base. This central storage contains multiple datasets, classifiers and performance values.

The datasets were selected mainly from the UCI machine learning repository [10]. We used 123 real-world datasets from different domains. Their sizes as well as their number and type of features differ.

Additionally, we selected a set of classifiers, that contains very successfully used algorithms (Random Forest, Neural Network, Support Vector Machine) as well as algorithms using different approaches (Naive Bayes, Decision Tree, k-Nearest Neighbor, One-R). Of course, there are many more classification algorithms, which can be easily added to the case base.

Every classifier was evaluated on each dataset. We determined the accuracy values of the classifiers by a grid search of the most important parameters using a ten-fold cross-validation. The resulting table of datasets, classifiers and performance values can be easily used for further comparisons.

B. Automatic Evaluation of Pattern Recognition Systems

The complete system for automatically evaluating pattern recognition pipelines was implemented as a RapidMiner plug-in so that it can benefit from RapidMiner in two ways. First, the system is easily usable especially for users who have worked with RapidMiner before. It uses the same interfaces and GUI elements as the standard RapidMiner software. Instead of manually writing configuration files in a format that the user needs to get used to, he can use the more intuitive graphical interface of RapidMiner.

Second, the system can easily benefit from the existing functionality of RapidMiner. Different modules of RapidMiner are used within the system and are available for testing it as well. For example, importing data sets into the system is very easy and powerful this way because it can already import any data format that is supported by RapidMiner. It is also easily possible to integrate the implemented plug-in into more complex processes.

The PaREn system is able to automatically evaluate a new classifier on all previously imported datasets. The classifier can be an arbitrary RapidMiner process that takes a dataset as input and delivers the learned model as output. Optionally, the user can specify a preprocessing pipeline that will be applied on every dataset before invoking the classifier itself.

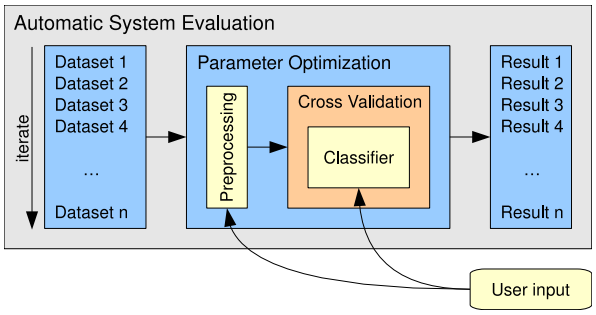


Fig. 2. Schematic overview of the automatic system evaluation process.

The actual evaluation of the classifier can be done in many different ways. Currently we use a cross-validation whereas the number of folds and the seed for the random generator are fixed parameters of each data set. The fixed seed is used for better reproducibility of the experimental results.

Additionally, for every classification pipeline, the user can define a set of parameters that should be optimized during its evaluation. The optimization methods are performed independently on each dataset and use a grid search.

If the user wants to evaluate a new classification pipeline including a preprocessing, the developed system automatically applies it on all datasets and measures the performance values using a cross-validation. The result is a list with the performance values of every data set that are stored in the central database. The overall process is illustrated in Figure 2. Using the system, it is easily possible to get an overview of the performance of a classifier over many different data sets. The comparison with other previously evaluated classifiers is much simpler as well.

All results are stored within a RapidMiner repository. This includes all datasets and for every evaluated classifier:

- the classifier process XML file
- the optional preprocessing XML file
- a list of parameters and their optimization intervals
- for every dataset:
  - the best parameter values found during optimization
  - the best performance value found during optimization
  - the complete automatically generated evaluation pipeline including preprocessing and parameter optimization setup

Since especially parameter optimization with cross-validation is a very compute intensive task, the software is able to do the evaluation on the datasets in parallel using multiple threads. It can also be integrated with the distributed approaches that we have developed (Section IV-D) to reduce computation times further.

C. Automatic System Optimization

The possibilities of RapidMiner for parameter optimization are rather limited. It offers implementations of a grid search and an evolutionary approach.

We implemented an operator for RapidMiner that uses Simulated Annealing for parameter optimization. Simulated Annealing may work faster than Evolutionary Algorithms since it does not use multiple solutions in parallel.

In order to overcome the complexity problem by not just using more hardware resources, we also implemented an adaptation of the Evolutionary Optimization operator of RapidMiner that is able to use defined parameter values as start population. By defining start points that are already promising by using the knowledge obtained from previous system evaluations, we can decrease the iterations, number of individuals and therewith the time needed for finding the optimal parameter values. This can be an advantage especially for users that are no experts and therefore cannot easily limit the search space of the parameter values. The risk of missing the optimal parameter set should also be reduced by this approach.

D. Dispare - Distributed Pattern Recognition

RapidMiner already provides easy to use interfaces for developing and evaluating Pattern Recognition and Machine Learning applications. However, it has only limited support for parallelization and it lacks functionality to spread long-running computations over multiple machines. A solution to this is distributed computing with paradigms like MapReduce.

In the PaREn project, we have developed and evaluated a system called *dispare* which integrates distributed computing frameworks into RapidMiner. A special focus is put on utilizing MapReduce as a programming model. The software framework of GridGain and Oracle Coherence is reviewed and evaluated with respect to its suitability to fit into the context of RapidMiner. The developed system provides effective means for transparently utilizing this framework and enabling RapidMiner processes to parallelize their computations within a distributed environment.

E. Collaborative User Interface

The *Collaborative User Interface* represents the graphical front-end to a rich set of software tools and datasets available as RapidMiner extensions. It allows users to access all major PaREn functionality easily over the web. This greatly reduces the required effort for beginners to access and comprehend pattern recognition technologies. A screen-shot of the user interface showing predicted accuracies of different classifiers is shown in Figure 3.

Users are able to upload data with the Collaborative User Interface for which they would like to have a classification pipeline. These pattern recognition pipelines are provided as RapidMiner processes in XML format. The Collaborative User Interface invokes the PaREn RapidMiner extensions for analysis of the uploaded data and keeps the user informed about the status of this process. Depending on the results of the data analysis accuracies of a pre-defined set of classifiers are predicted. The predicted accuracies are shown to the user and then he can pick one or more classifiers to be evaluated on his dataset to get actual accuracies. The PaREn

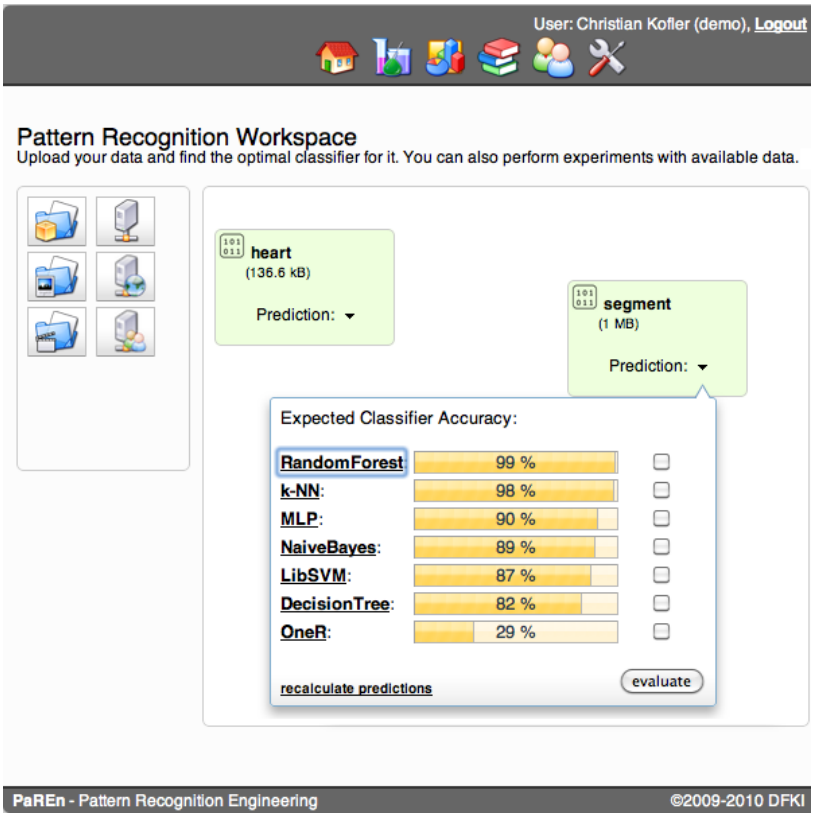


Fig. 3. A screen-shot of the collaborative user interface showing predicted accuracies of different classifiers on the user-provided dataset.

system then optimizes the chosen classifiers on the given dataset and provides a complete pattern recognition pipeline for download.

F. AutoMLP Classifier

In addition to automatic system evaluation and optimization, the PaREn goal of making pattern recognition systems accessible to non-experts requires development of new statistical methods and algorithms that are essential for reducing the number of parameters needed by pattern recognition and machine learning methods.

We started this work by creating a classifier for OCR applications that is self-configuring and self-tuning. Given training data (either supplied as a batch, or a sample at a time), the classifier will automatically select among a range of possible internal classifiers. The choice is determined by automated internal cross-validation and complexity measures, as well as available computational resources and memory. Classifiers can scale from a few training samples to millions of training samples.

After thorough testing and evaluation of self-tuning classifiers, we incorporated them in the 0.4 release of the OCRopus open source OCR system. A particularly promising classifier is the AutoMLP classifier, which automatically adjusts its learning rate and number of hidden units with an evolutionary approach. The implementation is done such that it can automatically cross-validate different samples in the population in parallel and converges to a good solution very fast. We are working now on applying it to general classification tasks and do extensive benchmarking against other standard classification algorithms. Since the original implementation was in C++ for performance reasons, we also provide a Java implementation of the AutoMLP classifier for RapidMiner.

G. Meta Feature Extraction

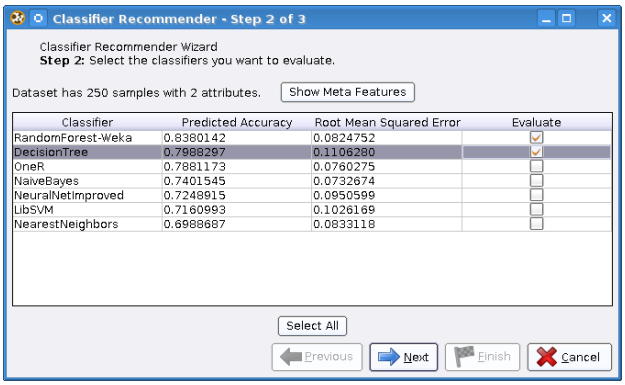
Meta-learning is a growing field in Machine Learning. Classifier recommendation, ranking of learning algorithms, or automatic classifier selection can be developed based on meta-features. Landmarking is a recent category of meta-features, which uses the accuracies of some simple classifiers as characteristics of a dataset. Considered as the first meta-learning step in RapidMiner, we have developed a new operator called *landmarking* to extract meta-features of a dataset. These meta-features can then be used to predict the accuracies of different classifiers on a dataset.

H. PaREn Automatic Classifier Selection Wizard

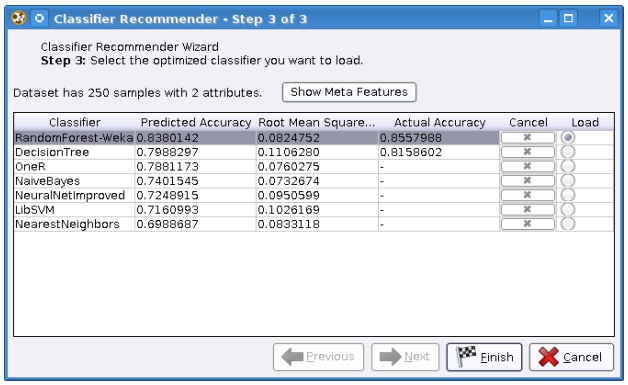
The wizard for automatic classifier selection guides the user in three simple steps to the best classifier for his dataset. The three steps are:

- Selection of the dataset
- Selection of the classifiers for evaluation
- Selection of the classifier for system construction

After the user selected the dataset he wants to get a pattern recognition system for, the data is analyzed and meta-features



(a) Second step: the user can select which classifiers should be evaluated based on the predicted accuracies.



(b) Third step: the user can select a classifier for the automatically constructed pattern recognition system.

Fig. 4. Screen-shots of the second and third step of the wizard for automatic classifier selection.

of it are calculated. Based on these features, the classification accuracy is predicted for every classifier of the wizard. Along with the accuracy values, the root mean squared errors (RMSE) for each classifier is shown. The RMSE indicates the confidence of the prediction.

Next, the user selects the classifiers, which should be optimized on the dataset. The decision is typically based on the predicted accuracy values. Classifiers with low predictions could be omitted in order to save computation time. For every selected classifier, a pattern recognition system including preprocessing is constructed automatically and important parameters are optimized. The optimization currently uses a grid search but can be easily replaced with a more sophisticated approach. The results of the optimizations are shown to the user.

As last step, the user can select the classifier for the pattern recognition system, that will be constructed automatically and loaded. Typically, this will be the system with the best performance, obtained by the previous optimization step.

The constructed system also includes preprocessing. For instance, it converts nominal to numerical features for algorithms that do not support nominal features. Since the optimal parameters found during optimization are already set, the constructed pattern recognition system can be used immediately.

V. Conclusion

In this paper, we presented several software results of the PaREn project. The RapidMiner based implementations support the user in developing, testing and comparing machine learning modules. Non-experts are able to use pattern recognition techniques much easier. The classifier selection wizard as well as the collaborative user interface reduces the amount of required expert knowledge. The performance of classification algorithms are predicted and complete pattern recognition system are constructed and optimized automatically. The AutoMLP classifier is a self-configuring multi-layer-perceptron and therefore increases the usability of this classifier especially for beginners. Its self-tuning approach also reduces the time needed for training and optimization.

Additionally, the automatic system evaluation and the collaborative user interface make it much easier to compare different pattern recognition systems. Using the Dispare extension for distributed computations can reduce the time of development and evaluation of such systems dramatically. The landmarking operator introduces the first meta-learning functionality to RapidMiner.

References

[1] Robert P. W. Duin, Fabio Roli, and Dick de Ridder. A note on core research issues for statistical pattern recognition. *Pattern Recognition Letters*, 23(4):493–499, February 2002.

[2] Theo Pavlidis. 36 years on the pattern recognition front: Lecture given at icpr'2000 in barcelona, spain on the occasion of receiving the k.s. fu prize. *ttern Recognition Letters*, 24(1-3):1–7, January 2003.

[3] R. King, C. Feng, and A. Shutherland. Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3):259–287, May/June 1995.

[4] Donald Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[5] Esprit project METAL (#26.357). A meta-learning assistant for providing user support in data mining and machine learning, 1999-2002. http://www.ofai.at/research/impml/metal/.

[6] Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 743–750. Morgan Kaufmann, 2000.

[7] Isabelle Guyon, editor. *Model Selection Workshop and Performance Prediction Challenge*, Vancouver, British Columbia, Canada, July 2006.

[8] Isabelle Guyon, Amir Reza Saffari Azar Alamdari, Gideon Dror, and Joachim M. Buhmann. Performance prediction challenge. In *International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre*, 2006.

[9] Gregory Piatetsky-Shapiro. Data mining tools used poll, 2009. http://www.kdnuggets.com/polls/2009/data-mining-tools-used.htm.

[10] C. J. Merz, P. M. Murphy, and D. W. Aha. UCI repository of machine learning databases. University of California, Department of Information and Computer Science, Irvine CA, 1997. http://www.ics.uci.edu/∼mlearn/MLRepository.html.