Predicting Execution Time of Machine Learning Tasks using Metalearning

Rattan Priya*, Bruno Feres de Souza†, André L. D. Rossi† and André C. P. L. F. de Carvalho†
*Computer Science Engineering
Indira Gandhi Institute of Technology, GGSIPU
New Delhi, INDIA
Email: bhasin.rattanpriya@gmail.com
†Computer Science Department
Insitutute of Mathematics and Computer Sciences, University of São Paulo
São Carlos-SP, Brazil
Email: bferes,alrossi,andre@icmc.usp.br

***Abstract*—Lately, many academic and industrial fields have shifted research focus from data acquisition to data analysis. This transition has been facilitated by the usage of Machine Learning (ML) techniques to automatically identify patterns and extract non-trivial knowledge from data. The experimental procedures associated with that are usually complex and computationally demanding. Scheduling is a typical method used to decide how to allocate tasks into available resources. An important step for such is to guess how long an application would take to execute. In this paper, we introduce an approach for predicting processing time specifically of ML tasks. It employs a metalearning framework to relate characteristics of datasets and current machine state to actual execution time. An empirical study was conducted using 78 publicly available datasets, 6 ML algorithms and 4 meta-regressors. Experimental results show that our approach outperforms a commonly used baseline method. Statistical tests advise using SVM_r as meta-regressor. These achievements indicate the potential of metalearning to tackle the problem and encourage further developments.**

***Keywords*-Machine Learning, Prediction time, Metalearning**

I. Introduction

Recently, the focus of research has shifted to a more qualitative perspective, considering not only data gathering and storage but studying and unearthing patterns and trends from available raw data. To couple with this challenge, Machine Learning (ML) [1] emerges as a valuable area that investigates how to develop and employ techniques that automatically identify patterns and extract novel and potentially useful knowledge from data.

Data analysis using ML techniques may involve very complex and time-consuming procedures, because there is a plethora of algorithms available, each one with different utility with respect to the requirements of the researchers and nature of the problem being investigated. Conducting the research relies on demanding empirical processes or expert advice [2]. For instance, practitioners tend to use only a few available algorithms for data analysis, hoping that the set of assumptions embedded in these algorithms will match the characteristics of the data in a trial-and-error manner. This is typically the case when one needs to perform an exploratory analysis to assess the behavior of a pool of algorithms on a batch of datasets.

To couple with the intensive computational requirements of ML applications, Distributed Heterogeneous Computing (DHC) systems [3] can be used. In a DHC environment, a suite of various resources with different capabilities, interconnected using high speed network, is coordinated to support the execution of a multitude of parallel and distributed tasks. A scheduling algorithm can allocate tasks into machines, thus optimizing the performance or cost-effectiveness of the system [4]. In this scenario, an accurate prediction of the execution time of these tasks provides essential information for the scheduler, reducing the response time for the workload [5].

Knowing the execution time and the resources usage for each task can be beneficial in two ways. First, scientists do not have to wait unexpectedly long for running their experiments. When these demanding tasks can be identified in advance, they can be scheduled without any resource contention. Second, accurate predictions may enhance the scheduling and thus minimize the overall execution time [6], [7]. There are many approaches to deal with this problem, like constructing analytical models based on simplistic assumptions that may not hold on real computing environments [8] and analyzing the internal semantics of the software, which are coding language specific [9]. An alternative approach was studied in [10], where the authors considered ML algorithms to build models able to predict the execution time of queries in a data warehouse.

In the present study, we approach the problem of predicting execution time of ML tasks over a DHC by applying regression models. The proposed methodology borrows key concepts from the Meta-learning literature [11]. For our purpose here, metalearning consists of using a learning algorithm to relate the characteristics of data and computational resources to execution time of ML tasks. It usually involves three steps: (1) the generation of meta-data; (2) the induction

978-1-4673-0126-8/11/$26.00

of a metalearning model by applying a regression algorithm to the meta-data and; (3) the application of the meta-model to predict the running time of ML tasks. By ML applications we mean the employment of multiple ML algorithms over multiple learning problems (datasets). Each task of the application corresponds to the execution of an algorithm over a dataset. It represents a very common real world situation where various techniques need to be evaluated and compared in a comprehensive set of problems (e.g. see [12]).

The text is organized as follows. Section II discuss some related work in predicting execution time. The methodology proposed in this paper to deal with this problem is described in Section III. Section IV presents the experiments performed using our methodology and discuss the achieved results. Finally, Section V provides the concluding remarks and discusses future research directions.

II. Prediction of application execution time

Intensive scientific applications have increased the need for using a large number of computational resources. If the number of tasks is larger than the number of machines, a scheduling mechanism to optimize the usage of DHC over time [7], [6] is necessary. An important issue to improve scheduler performance is accurately predict the execution time of these tasks. According to [9], experts usually have to perform detailed analysis on algorithms and select the most suitable features to build prediction models for this problem. Consider, for example, a simple situation where an environment have two machines available, one has a CPU with high processing capacity, named C_1, and the other is slower, named C_2. In addition, consider that one wants to execute two programs, namely P_1 and P_2, and both have to meet time-constraints. If it is possible to predict execution time of each program for each machine with reliable accuracy, it would be possible to obtain better resource utilization deciding if both tasks would execute in C_1 or if one task would execute in C_1 and the other in C_2.

In contrast to the demand to optimize computational resources, [9] state that prediction systems have been at best unsophisticated, considering the advances in many computer areas. One example is the analytical models based on simplistic assumptions [8]. In order to deal with this problem, [9] propose a new system able to automatically extract a number of features from program execution that are used to construct predictive models using Sparse Polynomial Regression (SPORE) without expert knowledge. These features include information regarding the internal variables, loops and branches. Hence, this system tries to learn a relationship between a response (e.g., the execution time of a machine program) and these features to build an explicit polynomial model. The SPORE methodology deals with non-linear effects of the features by using polynomial basis functions over each feature.

The problem of predicting the execution time of a query on a loaded data warehouse with a dynamically changing workload is investigated in [10]. The authors propose an approach, namely Predicting Query Run-time (PQR) Trees, to predict the query execution time in the form of time ranges. This approach uses characteristics of historical data from several queries and system load under varying conditions to build a predictive model.

A method based on the K-Nearest Neighbor algorithm is used in [13] to predict program running time based on static features, such as user_id and scheduling queue number, and dynamic attributes, like the amount of memory used, the number of processing cores and the CPU time. The knowledge acquisition model is based on the observation that the most similar the programs the highest the chance of presenting close execution times.

According to [6], application execution times are difficult to predict, mainly in shared environments that run parallel applications because computational resources capacities (e.g., CPU load, bandwidth, latency) can change dramatically over time. To deal with this problem, they used regression models to predict the execution time of parallel applications in shared environments. The regression models were constructed by using past historical data. However, they did not consider parallel applications that are nondeterministic regarding the execution time or that depend on data distribution.

Three techniques based on regression are investigated in [5] for predicting parallel program scalability. The authors performed several executions with different datasets using a small subset of processors. The first technique uses the relationship between the gathered data on training runs to develop a predictor for a larger number of processors. The other two techniques are extensions of the first and handle computation and communication separately.

III. Metalearning for time prediction of ML tasks

As previously mentioned, regression models have been successful used to predict the execution time of applications. Motivated by these ideas, in this paper we investigate an approach that uses some concepts of the metalearning [14], [11] area to build regressors models to predict the execution time of classification algorithms. The training of the learning algorithms uses a meta dataset, which is created from the current system load and extracting characteristics from the datasets. These characteristics are computed using measures that have been extensively studied in metalearning. The information about the current computational state are obtained using simple commands from Linux operating system.

In our approach, regression models are induced offline for each combination of type of machine, ML algorithm, and dataset to predict reliable estimates of the execution times. It is termed offline because if a new type of computer or

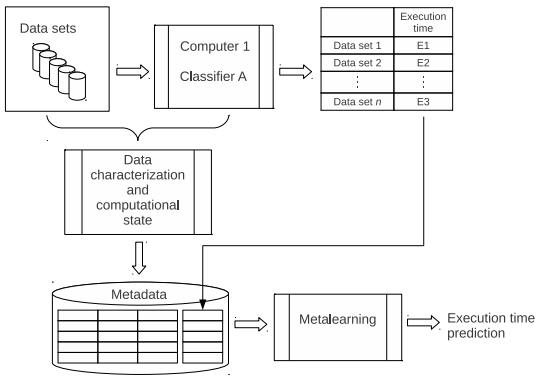

Figure 1. Metalearning approach for predicting execution time of ML algorithms.

algorithm is added to the distributed system, new experiments need to be run for the datasets in order to estimate new running times.

Figure 1 shows a diagram of the approach proposed here, which can be divided in two main parts: 1) The characterization of the datasets and computational resources; 2) Induction of the regression models. First, a learning algorithm is trained for different datasets, inducing different models under a specific machine. The information about the execution time that a particular algorithm took to run each dataset and the current status of the machine during the process are stored. Then, the characterization component extract the relevant features from the datasets and the information about the machine status that were stored to create a meta-data. Each line of this meta dataset is an example, and each column is an attribute, whose values were gathered from the training of the learning algorithms, where the last column is a special attribute namely target that is the execution times. In the second part, these meta-data are used to induce a meta-regressor algorithm aiming to discover the relationship between the attributes values (independent variables) and the execution times (dependent variable). The model have to be able to generalize for different datasets, using the same learning algorithm under the same computational environment.

Once the model is created using the meta-data, it is necessary to produce evidence to the user that the meta-regressor is able to generate accurate predictions. An approach for this is to use Leave-one-out Cross Validation (LOOCV), which iteratively, for each meta-example, computes the classifier execution time using a meta-model obtained on all the remaining meta-examples [11]. Other methods, such as the 10-fold cross validation, could be used, but the LOOCV generates more reliable estimates, mainly when there are a small number of examples, that it is the case here. The accuracy of the predictions was assessed according to the Mean Absolute Deviation (MAD). It is defined as the sum of the absolute differences between actual and predicted execution times divided by the number of test items in the LOOCV. To determine whether the accuracy of some particular predicted value can be regarded as high or not, a baseline method is required. Usually, simple prediction strategies that summarize the values of the target variable for all examples in the dataset are employed. For regression, a common baseline is established by default MAD (dMAD), which is obtained by predicting the test target using the average targets of the training sets. Since average values are very sensitive to outliers, we consider here dMAD computed using the median of the targets.

A. Metadata design

The characteristics that constitute the attributes of the meta data are extracted from the datasets and from the current status of the computational resources. The former provides static measures whereas the latter are dynamics. There are several measures in the metalearning area that were proposed to characterize datasets [15]. Several studies in this area concerns about ML algorithms recommendation, and some of them, like in [16], includes the execution time as an important component for this task. In our study, the following measures were used to extract the datasets characteristics [16]:

- INST: Log of number of instances;
- ATTR: Log of number of attributes;
- CLAS: Log of number of classes;
- PROP: Proportion of continuous attributes;
- ENTR: Normalized class entropy;
- CORA: Average absolute correlation of all pairs of attributes;
- CORT: Average absolute correlation between target and attributes.

Other measures are related to the current state of the computational environment, as follows:

- MEM: Current free memory available in the machine;
- CPU: Current CPU idle in the machine.

These measures are computed and used to constitute the metadata, where each measure is an attribute and the target of each instance is the execution time.

IV. Experiments

In this section we present the experiments performed to evaluate the proposed approach. The aim here is to predict the execution time of a ML application with several tasks in a DHC environment. Due to space constraints, experiments results reported here employ just one type of machine. The full set of results considering all machines can be retrieved from the Supplementary material in www.icmc.usp.br/~bferes.

A. Experimental settings

In order to perform the experiments we used 78 classification datasets that were obtained from the UCI Machine Learning Repository [17]. These datasets were used to train six classification algorithms available in WEKA [18], namely: K-Nearest Neighbor (K-NN$_c$), Support Vector Machine (SVM$_c$), Decision Tree (J48), Rules System (JRip), Bagging and Naive Bayes (NB). They represent distinct ML paradigms and have different inductive bias. Thus, they are expected to be representative of commonly used ML algorithms. Execution time of each classifier is the sum of five runs, and this value in seconds is used in the following experiments. The following regressors were used to predict the classifiers execution time: Linear Regression (LR), Decision Tree (M5P), K-Nearest Neighbor (K-NN$_r$) and Support Vector Machines (SVM$_r$). All these techniques were already used in previous work as meta-regressors in other contexts (see, for instance, [11]). Classification and regression algorithms employed here were executed using their default parameter values. Details about, datasets, algorithms default parameters and machine configurations can be retrieved from Supplementary material.

B. Experimental results

The quality of the estimation of the four regression models using the available metadata for the six classifiers is shown in Table I. It presents the MAD values for LOOCV for every regressor and the dMAD. The best value for each classifier is highlighted. Lower MAD indicates better performance of the regressor. As can be noted, the majority of the regressors are able to consistently generate average predictions more accurate than the baseline method. SVM$_r$ and K-NN$_r$ present the best overall performance, followed by M5P. In contrast, LR predictions are worse than dMAD for four classifiers. Stratifying results by classifiers, execution time of K-NN$_c$ is relatively harder to predict for all regressors and dMAD. Conversely, accurate estimates for NB are straigthforward to achieve. Such behaviors may be related to: 1) unusual performance of the classifiers or 2) unsuitable set of meta-features used .

Table I
MAD OF THE REGRESSORS CONSIDERING ALL META-FEATURES.

	K-NN$_c$	**SVM$_c$**	**J48**	**JRip**	**Bagging**	**NB**
LR	8.105	5.011	1.264	3.174	7.125	0.615
M5P	5.394	4.343	**0.784**	**2.391**	4.764	0.573
K-NN$_r$	5.090	3.384	0.886	2.755	3.925	0.579
SVM$_r$	**4.905**	**3.180**	0.823	2.684	**3.773**	**0.481**
dMAD	6.815	3.786	1.125	3.515	4.466	0.666

Figure 2 indicates that the first issue may lead, to a certain degree, to drops in performance. It shows the boxplots of the distribution of the execution times for the 6 classifiers on the 78 datasets used here. J48 and NB boxplots have comparatively smaller heights, which can explain why MAD

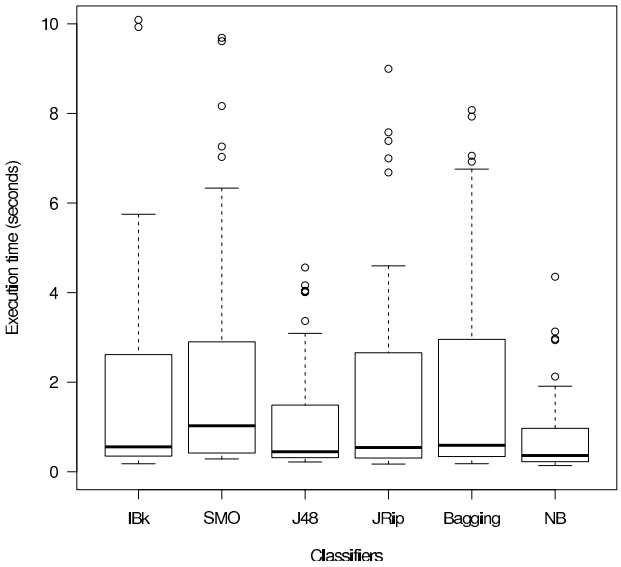

Figure 2. Boxplot of the distribution of actual execution times.

values for these classifiers are better than the others. On the other hand, K-NN$_c$, SVM$_c$, JRip and Bagging boxplots exibits a wider range of values, which makes the execution times more difficult to predict. In addition, all classifiers present many outliers. Some of them are not showed in the plot, because they far exceed 10 seconds. The most deviant case occurs for K-NN$_c$ classifier with the execution time of 160.81 seconds for SECOM dataset. Such atypical values adversely impact regressors performance, which is reflected by high MAD entries in Table I.

In order to assess how suitable the employed meta-features are for the problems at hand we compute Pearson correlation for each pair of classifier and meta-feature. As can be seen in Table II, values are usually not very high. INST and ATTR are the most correlated ones for all algorithms. They are used to measure how scalable a classifier is regarding the size of its input. In general, these datasets characteristics play important role in the computation time, so it is expected that the higher the amount of data, the higher the processing effort required. Other static meta-features present smaller but apparent influence on the response output, namely PROP, CORA and CORT. ENTR and CLAS, which express class information, are not strong. Considering dynamic measures, MEM seems to be relevant whereas CPU does not. Since the machine used in the experiments conducted here has high raw power, this could have minimized the importance of CPU current state. On the contrary, total free memory available negatively correlates with the target. In order to determine wheather the correlation coefficients are statistically meaningful, we applied an appropriate t-test [19]. Highlighted entries correspond to significant values.

Once a preliminary screening of meta-features was per-

Table II
PEARSON CORRELATION AMONG META-FEATURES AND CLASSIFIERS.

	INST	ATTR	CLAS	PROP	ENTR	CORA	CORT	MEM	CPU
K-NN$_c$	**0.43**	**0.49**	-0.10	0.03	-0.16	-0.17	-0.18	-0.14	0.03
SVM$_c$	**0.28**	**0.25**	0.08	**-0.24**	-0.05	-0.13	-0.16	-0.11	-0.12
J48	**0.40**	**0.60**	-0.03	0.00	-0.20	-0.20	-0.20	**-0.20**	-0.03
JRip	**0.62**	**0.45**	0.14	0.18	-0.11	-0.04	-0.13	**-0.33**	0.03
Bag	**0.23**	**0.27**	-0.03	-0.17	0.04	-0.14	-0.17	-0.21	0.05
NB	**0.37**	**0.44**	0.04	-0.17	0.01	-0.16	-0.19	**-0.33**	-0.13

formed, we could identify which of those are potentially more suitable to describe our problem. Considering only the measures with significant correlation for at least one classifier (Table II), a new round of experiments was carried out. In Table III, one can realize that results generally improved with the more compact set of meta-features. The best value for each classifier is highlighted. Crossing information of both tables, lower MADs for K-NN$_c$, J48 JRip, NB match the expectation, since less relevant meta-features for those classifiers were removed. For SVM$_c$, MADs presented minor variations, being better for LR and M5P and worse for K-NN$_r$ and SVM$_r$ regressors. This indicates that some of the removed meta-features may be useful or not, depending on the induction algorithm. Similar behavior was presented by Bagging.

Table III
MAD OF THE REGRESSORS CONSIDERING SELECTED META-FEATURES.

	K-NN$_c$	SVM$_c$	J48	JRip	Bagging	NB
LR	7.241	4.907	1.095	3.101	7.102	0.615
M5P	4.845	4.305	0.762	**2.214**	4.862	0.515
K-NN$_r$	**4.053**	3.477	0.734	2.700	**3.480**	0.510
SVM$_r$	4.658	**3.192**	**0.691**	2.378	3.715	**0.415**
dMAD	6.815	3.786	1.125	3.515	4.466	0.666

The outcomes presented till now are the MAD values of the predictive models. In order to provide more evidence that the observed differences between each regressor and dMAD are significant, we applied a t-test as suggested by [20]. The null hypothesis to be verified is the equality between their errors. The resulting p-values are showed in Table IV. Since various statistical comparisons are being performed, Bonferroni correction is employed [21]. Thus for an overall confidence level of 95%, we should have $\alpha = 0.00208$. Values smaller than α are highlighted. As can be noted SVM$_r$ is significantly better than dMAD for five out of six classifiers. K-NN$_r$ is the second best regressor with statistical significant values for K-NN$_c$, J48 and Bagging. In contrast with these regressors, LR is statistically worse than dMAD for Bagging classifier. Although the M5P presents smaller MAD values (Table III) than dMAD for most of the classifiers, their errors are not significantly different. Stratifying analysis by classifier, p-values indicate that at least one regressor could statistically outperform dMAD for five out of six classifiers. On the other hand, none regressor is significantly better than dMAD for JRip.

Table IV
P-VALUES OF THE PAIRED t-TEST WITH 0.1 OF SIGNIFICANCE.

	K-NN$_c$	SVM$_c$	J48	JRip	Bagging	NB
LR	0.7074	0.0127	0.8594	0.4684	**0.0011**	0.5466
M5P	0.0250	0.2258	0.0263	0.0158	0.5111	0.0577
K-NN$_r$	**0.0023**	0.3589	**0.0012**	0.1791	**0.0004**	0.0223
SVM$_r$	**0.0025**	**0.0029**	**0.0001**	0.0057	**0.0010**	**0.0000**

V. CONCLUSIONS

Nowadays, different ML techniques have been used to automatically extract useful information from the massive amount of data that has been generated by several technologies. The experimental procedures involved in such investigations usually are complex and time consuming, suggesting they would be more efficiently coupled allocating tasks of ML applications into different sets of computational resources. In this context, an important feature of a typical scheduler is to guess how long an application would take so a suitable schedule for the tasks could be planned. The major contribution of this paper is to explicitly cast the issue of predicting execution time as a metalearning problem. In the proposed design, regression models are used to relate characteristics of datasets and machines current state to execution time of classification algorithms.

The experiments carried out here considered predicting time of six classifiers over 78 UCI datasets using four regression algorithms. We compared how accurately they perform against a commonly used baseline. In Section II we can observe that our approach achieves promising results, usually yielding MADs smaller than dMAD method. The lowest average MAD values were obtained by SVM$_r$ and K-NN$_r$ regressors. Regarding classifiers, execution time for NB is the easiest to predict, leading to MADs lower than one second. On the contrary, predicting the behavior of K-NN$_c$ is not trivial. Complementary analyses using statistical test show that SVM$_r$ is the most advised meta-regressor, and that regressors were unable to significantly outperform dMAD for JRip.

Although the correlations among meta-features and response outputs were generally not high, selecting a subset of them improved the results for the majority of the cases. It is important to note that the correlation using Pearson coefficient to evaluate the importance of meta-features has some drawbacks, such as its plain linearity and univariate analysis. Thus, as future work, we intend to investigate more sophisticate feature selection methods.

Other possible research direction would be to investigate different sets of meta-features, both static and dynamic ones. In order to characterize datasets, one can use more sophisticated measures from metalearning, specifically those employed to analyze classification complexity [11]. Moreover, by now, regression models are classifier and machine dependent. In order to generalize our approach for predicting execution time, we intend to verify the suitability of using

multiple regression models (e.g. Partial Least Square [22]) to deal with several classifiers at once and considering static features of machine (e.g. processor type, amount of physical memory, etc) to simultaneously couple with different configurations.

ACKNOWLEDGMENT

The authors would like to thank the financial support of funding brazilian agencies FAPESP, CAPES, and CNPq, and the Department of Science and Technology, Government of India.

REFERENCES

[1] T. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, March 1997.

[2] A. Kalousis and M. Hilario, "Representational issues in meta-learning," in *ICML*, T. Fawcett and N. Mishra, Eds. AAAI Press, 2003, pp. 313–320.

[3] T. D. Braun, H. J. Siegel, and A. A. Maciejewski, "Heterogeneous computing: Goals, methods, and open problems," in *Proceedings of the 8th International Conference on High Performance Computing*, ser. HiPC '01. London, UK: Springer-Verlag, 2001, pp. 307–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=645447.652910

[4] H. Topcuouglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, pp. 260–274, March 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=566137.566142

[5] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. de Supinski, and M. Schulz, "A regression-based approach to scalability prediction," in *Proceedings of the 22nd annual international conference on Supercomputing*. New York, NY, USA: ACM, 2008, pp. 368–377.

[6] B.-D. Lee and J. M. Schopf, "Run-time prediction of parallel applications on shared environments," *Cluster Computing, IEEE International Conference on*, pp. 487–492, 2003.

[7] S. S. Vadhiyar and J. J. Dongarra, "A metascheduler for the grid," in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 343–351.

[8] S. Goldsmith, A. Aiken, and D. S. Wilkerson, "Measuring empirical computational complexity," in *ESEC/SIGSOFT FSE*, I. Crnkovic and A. Bertolino, Eds. ACM, 2007, pp. 395–404.

[9] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and M. Naik, "Predicting execution time of computer programs using sparse polynomial regression," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., 2010, pp. 883–891.

[10] C. Gupta, A. Mehta, and U. Dayal, "Pqr: Predicting query execution times for autonomous workload management," *Autonomic Computing, International Conference on*, vol. 0, pp. 13–22, 2008.

[11] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*. Springer Verlag, 2009.

[12] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," in *Proceedings of the 25th international conference on Machine learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 96–103. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390169

[13] L. J. Senger, M. J. Santana, and R. H. C. Santana, "An instance-based learning approach for predicting execution times of parallel applications," in *Proceedings of the 3rd International Information and Telecommunication Technologies Symposium*, 2005, pp. 9–15.

[14] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligent Review*, vol. 18, no. 2, pp. 77–95, 2002.

[15] C. Soares, "Learning rankings of learning algorithms: Recomendation of algorithms with meta-learning," Ph.D. dissertation, Faculdade de Ciências da Universidade do Porto, Porto, Portugal, 2004.

[16] P. B. Brazdil, C. Soares, and J. P. da Costa, "Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results," *Machine Learning*, vol. 50, pp. 251–277, 2003.

[17] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explorations Newsletter*, vol. 11, pp. 10–18, November 2009.

[19] R. R. Sokal and F. J. Rohlf, *Biometry*. W. H. Freeman and Co.: New York, 1995.

[20] H. Bensusan and A. Kalousis, "Estimating the predictive accuracy of a classifier," in *Machine Learning: ECML 2001*, ser. Lecture Notes in Computer Science, L. De Raedt and P. Flach, Eds. Springer Berlin / Heidelberg, 2001, vol. 2167, pp. 25–36.

[21] J. M. Bland and D. G. Altman, "Multiple significance tests: the bonferroni method," *British Medical Journal*, vol. 310, no. 6973, p. 170, 1986.

[22] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica Chimica Acta*, vol. 185, no. 0, pp. 1–17, 1986.