The final publication is available at IOS Press through http://dx.doi.org/10.3233/JIFS-169141

A meta-learning based framework for building algorithm recommenders: An application for educational arena

Diego García-Saiz* and Marta Zorrilla
Department of Computer Science and Electronics, University of Cantabria, Santander, Spain

Abstract. The task of selecting the most suitable classification algorithm for each data set under analysis is still today a unsolved research problem. This paper therefore proposes a meta-learning based framework that helps both, practitioners and non-experts data mining users to make informed decisions about the goodness and suitability of each available technique for their data set at hand. In short, the framework is supported by an experimental database that is fed with the meta-features extracted from training data sets and the performance obtained by a set of classifiers applied over them, with the aim of building an algorithm recommender using regressors. This will allow the end-user to know, for a new unseen data set, the predicted accuracy of this set of algorithms ranked by this value. The experimentation performed and discussed in this paper is addressed to evaluate which meta-features are more significant and useful for characterising data sets with the end goal of building algorithm recommenders and to test the feasibility of these recommenders. The study is carried out on data sets from the educational arena, in particular, targeted to predict students' performance in e-learning courses.

Keywords: Meta-learning, regression, student performance, educational data mining

1. Introduction

In this new era of so-named datification [26], there is an urgent need of both data scientists and tools addressed to make data management and analysis easier. Even though new technologies are arising in the data mining field, the automation of the Knowledge Discovery Process (KDD) is still an open problem. As it is well-known, the KDD process [28], which aims at identifying valid, novel, potentially useful, and ultimately understandable patterns in data, comprises several phases: preprocessing, modeling, mining and testing, and each one, in turn, includes a large number of tasks which must be performed. Most of these steps can be chained by means of scientific workflows but the choice of the most suitable techniques and algorithms to be utilised is not yet automatised. In particular, this work is a step towards the dynamic selection of the classifier to be applied on a certain data set at hand. As derived from the *no-free-lunch theorem*, no learning algorithm outperforms better than others for the set of all real-world problems [10], mechanisms that help data miners to select the most appropriate technique must be therefore developed.

Rice [15] was the one who first formulated this issue and since then different approaches have been proposed, for instance: a) a traditional approach based on a costly trial-and-error procedure; b) the use of ensemble methods to obtain better predictive performance; or c) an approach based on meta-learning, able to automatically provide guidance on the best alternative from a set of meta-features.

The proposed framework follows this last approach since the characterisation of data sets and the analysis of the behaviour of different machine learners applied on them have shown to be suitable and efficient (see Section 1). Although there are several works about meta-learning, only a few have focused on the algorithm recommendation and even fewer have studied which meta-features are the most suitable for this goal. This paper thus advances in both research lines with the following contributions:

- The explanation of how to build algorithm recommenders supported by an experimental database.
- The description of the set of meta-features that can be used for data sets' characterization.
- A sounded case study that analyses the behaviour of different sets of meta-features on data sets from the educational arena and their suitability for the building of algorithm recommenders.

This framework could be well utilised by practitioners in order to create and feed their own experimental database and use it as benchmark [16] as well as to build recommenders that help non-expert data miners to take advantage of mining techniques but hiding their choice and setting [19]. This is the case of teachers involved in virtual education, that due to the lack of face-to-face contact, require to analyse the activity performed in the e-learning platform to guide learners [9].

This work is a widely extended version of [19] in which the framework and the experimental database have been more clearly described and a new experimental study has been carried out using a higher number of algorithms and some different data sets in the training phase. Moreover, instead of one, two algorithms recommenders have been built and next compared to draw conclusions about the feasibility of the proposal, one based on linear regression and the other one on Multilayer Perceptron. Related work has been also reviewed.

This paper is organised as follows: Section 2 briefly describes the different elements which comprise our framework, previously introducing the meta-learning field and the set of meta-features that can be used. Section 3 describes the methodology followed in the case study and the setting of the experiments performed. Section 4 presents and discusses the results obtained showing the feasibility of the proposal. Finally, conclusions and future works are outlined in Section 5.

2. Background and an overview of our framework

Meta-learning is a subfield of machine learning which aims at applying learning algorithms on meta-features extracted from machine learning experiments in order to better understand how these algorithms can become flexible in solving different kinds of learning problems, hence to improve the performance of existing learning algorithms [25] or to assist the user to determine the most suitable learning algorithm(s) for a problem at hand [1], among others.

In this paper, meta-learning is used with the aim of learning the relationship between the meta-features extracted from the data sets and the algorithms performance applied on them. Therefore, a meta-learning system consists of two main stages: a training phase and a prediction phase. In the training stage, data sets are first characterised by a set of measurable meta-features and then, a set of classifiers are executed on them. The performance of these models generally measured by their accuracy (although other measures can be used such as f-measure, error rate, etc.) is next linked to the meta-features of each data set involved. Later, a learning algorithm is trained on the collected meta-features yielding a model which will be used to predict which the best algorithm to be applied on a new data set is. On other occasions, instead of selecting an algorithm, a ranking of algorithms is provided [7]. Different approaches for building the predictor are found, mainly based on classification [18, 21, 22] and regression [5, 23].

Recently, a new approach called meta-learning template [24] has arisen with the aim of recommending a hierarchical combination of algorithms. On the other hand, Britto et al. [2] studied the classification problem complexity using multiple classifier systems based on the dynamic selection of classifiers.

Regarding the type of meta-features, many have been proposed and applied, being commonly categorised in:

- Simple meta-features, such as the number of attributes, the number of instances, the type of attributes (numerical, categorical or mixed), the number of values of the target attribute and dimensionality of the data set, i.e., the ratio between the number of attributes and the number of instances.
- Statistical meta-features, like skewness, kurtosis among others which characterise data distribution [25, 27].

– Information theoretic meta-features used for characterising data sets containing categorical attributes such as class entropy or noise to signal ratio [20].
– Model-based meta-features, which collect the features of the mining model built, for instance, the structural shape and size of a decision tree trained on the data sets [30].
– Landmarkers, which are meta-features calculated as the performance measures achieved by using simple classifiers [4].
– Complexity meta-features, that characterise the apparent complexity of data sets for supervised learning. These are provided by DCoL (data complexity library) [29]. They have been used in several metalearning works [7, 12, 13, 18] and, recently, Herrera et al. [14] applied them to obtain the domains of competence of a classifier, which allows to predict if any data set will be suitable for such learning method or not.
– Contextual meta-features, i.e., characteristics related to data set domain [7, 18].

For the sake of a better understanding, a modular schema of the proposed framework is depicted in Fig. 1. As can be observed, this framework basically makes use of four workflows in the training phase: one for extracting meta-features of the data sets (WF1); another one, for generating models with each classifier under study (WF2); a third one, responsible for loading the descriptive information of each experiment performed on each data set into the database (WF3); and the fourth one, in charge of building a regressor for each type of algorithm used in the training phase establishing as class the value of the type of measure desired (accuracy, f-measure, and so on) (WF4), being accuracy the most frequently utilised [6]. Regarding the predictive phase, only a workflow is required. This is responsible for reading each new data set, extracting its meta-features, applying this meta-data set to the regressors previously built and showing the algorithms ranked according to the value of accuracy (or whatever evaluation measure chosen) predicted by themselves.

The database schema used to gather the experiments is depicted in Fig. 2. This was designed based on the one proposed in [16] which collects machine learning experiments. But this schema had to be extended to store meta-features extracted from each data set (Fields and DatasetMetaFeatures classes) and the set of meta-features which describes each mining model built (MiningModels and Measures classes).

2.1. Meta-learning in educational arena

Meta-learning is a subfield of machine learning which has not been yet well-explored to be applied

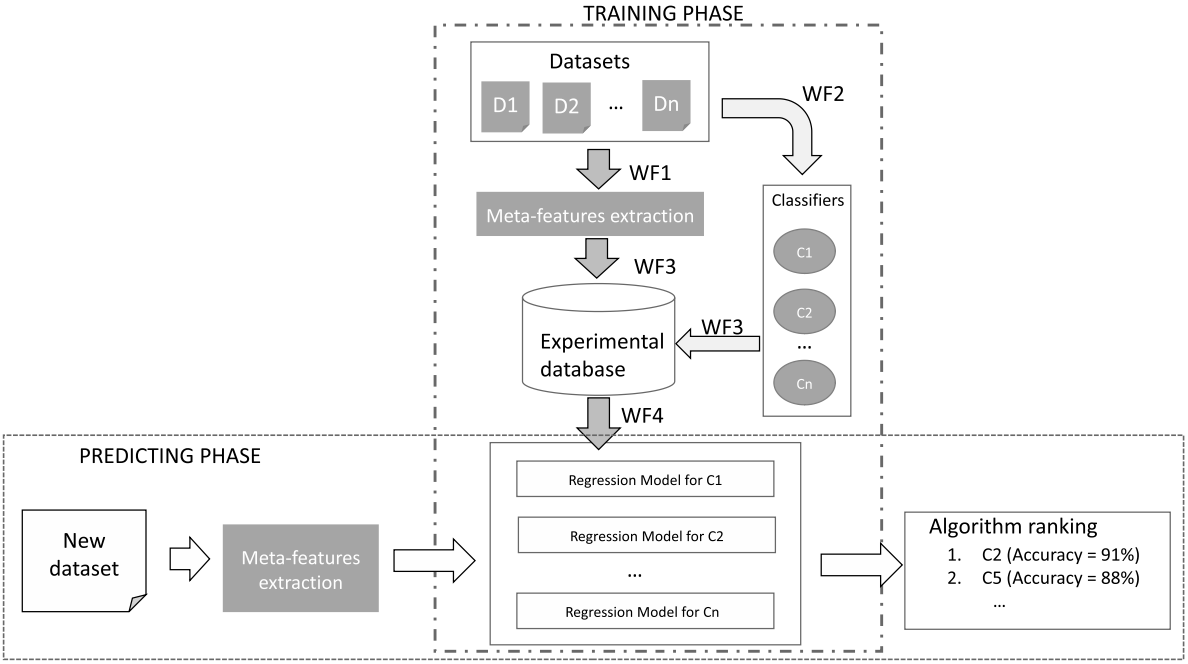

Fig. 1. Overview of the metalearning framework proposed.

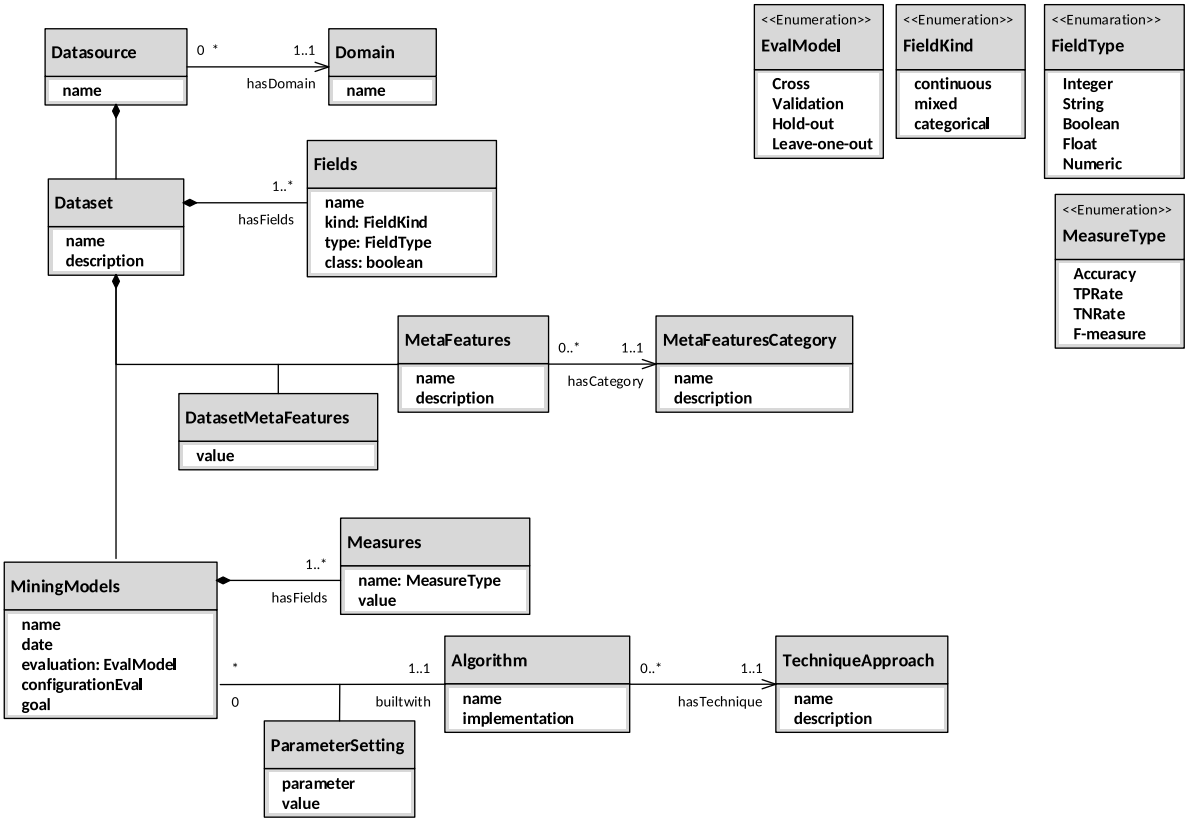

Fig. 2. UML database schema for storing meta-features of mining experiments.

on the educational arena, despite the huge quantity of data available in learning platforms and also the urgent need of analysing and improving the learning processes to improve the academic performance and avoid the worrying dropout. Although many issues have been addressed according to these surveys [3, 8, 9], few works relied on a metalearning based solution. Romero et al. [21] proposed the use of metalearning for the automatic setting of two parameters of the J48 algorithm with the aim of increasing the model accuracy for predicting student's performance. Later, this research group built a recommender for selecting the best classifier using a nearest neighbor (1-NN) approach on statistical, complexity and domain meta-features [7] and next, applied a multilabel learning algorithm [13] for the same goal. On the other hand, Zorrilla et al. [19] evaluated the possibility of building a recommender to be wrapped in a data mining service using only the classifier that outperformed in the training phase of the meta-learning process. The results were positive but limited because the number of algorithms could not be increased due to the reduced number of training data sets available. Some authors like [17] solved this issue first clustering algorithms based on behaviour similarity and, then recommending a set of algorithms, instead of an algorithm whereas others followed an approach based on regression [19, 23]. According to Lemke et al. [6], this last approach has been barely studied, thus the experimental study carried out in this work is addressed to fill this gap.

3. Experiment design

This mining experiment aims at studying the relevance and advantages that each group of meta-features provides for the building of algorithm recommenders as well as assessing the predictive power of these recommenders. The process followed is the one detailed in Fig. 1.

All data sets included in the experiment came from thirty different blended and virtual learning courses hosted in a Moodle platform. These gather the activity carried out by the students measured by means of metrics defined at course level and at tool level

which are available in the course such as the total number of sessions opened by each student in the course, time spent per session, number of self-tests performed, number of messages posted and answered in the forum, among others. All attributes are numeric except the class attribute which collects whether the learner failed (positive class) or passed (negative class) the course.

Once training data sets were loaded, their meta-features were extracted. Concretely, the following ones were used: a) the number of instances, the number of attributes and the dimensionality as simple meta-features; b) the minimum, the maximum and the average value of the skewness and kurtosis of all attributes of the data set calculated as statistical meta-features, by means of the MATH3-apache Java library; c) the fourteen complexity meta-features offered by DCoL software [29] that are, the maximum Fisher's discriminant ratio (F1), the directional-vector maximum Fisher's discriminant ratio (F1v), the overlap of the per-class bounding boxes (F2), the maximum (individual) feature efficiency (F3), the collective feature efficiency (sum of each feature efficiency)(F4), the fraction of points on the class boundary (N1), the ratio of average intra/inter class nearest neighbor distance (N2), the training error of a linear classifier (N3), the fraction of maximum covering spheres (T1), the average number of points per dimension (ratio of the number of examples in the data set to the number of attributes)(T2), the leave-one-out error rate of the one-nearest neighbor classifier (L1), the minimized sum of the error distance of a linear classifier (L2) and the nonlinearity of a linear classifier (L3); d) the accuracy achieved by the following weak classifiers as landmakers: LinearDiscriminant (LD), BestNode with gain-ratio criterion (BN), RandomNode (RN), NaïveBayes (NB) and 1-NN, all available in Weka or RapidMiner.

As a consequence of the fact that the data sets only had numeric features, no information-theory measures were used. Likewise, due to the great variability of algorithms used in the training phase, model based meta-features were discarded. Table 1 shows that the meta-features extracted from training data sets take a wide range of values.

Next, eighteen classifiers from different approaches were run on these thirty data sets using their default setting. The implementations chosen were AdaBoostM1, ADTree, Bagging, BayesNet, BFTree, J48, Jrip, Logistic, MultiBoostAB, MultilayerPerceptron, NNge, OneR, RandomForest, Ridor, SimpleCart, SMO and VotedPerceptron, all of them available in Weka. The accuracy achieved by each classifier on each data set was stored in the data base. The validation process followed was leave-one-out.

Table 1
Range of values of the meta-features

Type	Name	Range
Simple	N# attributes	2-21
	N# instances	13-502
	dimensionality	0.025-0.647
Statistical	skewness-avg	(-)1.48-4.69
	skewness-min	(-)8.124-3.227
	skewness-max	1.425-18.01
	kurtosis-avg	1.42-50.83
	kurtosis-min	(-)1.51-15.22
	kurtosis-max	4.56-339.88
Complexity	F1	0.09-7.26
	F1v	0.00-387.84
	F2	0.00-0.16
	F3	0.02-0.88
	F4	0.03-1.00
	L1	0.24-0.97
	L2	0.09-0.47
	L3	0.05-0.50
	N1	0.05-0.65
	N2	0.25-0.94
	N3	0.00-0.40
	N4	0.00-0.36
	T1	0.60-1.00
	T2	1.70-47.25
Landmarkers	BestNode	1.54-87.88
	RandomNode	40-97.50
	NaiveBayes	48.96-100
	LinearDiscriminant	23.33-97.50

Then, eighteen meta-data sets were generated, one for each classifier. Each data set contained the meta-features of the training data sets along with the accuracy achieved by that specific classifier. For the sake of studying the behaviour of each group of meta-features, we built different linear regression models by using different combinations of meta-features:

1. Using all the meta features available.
2. Using only the meta-features which belong to each group (simple, statistical, complexity or landmarkers) separately.
3. Using only the most relevant meta-features chosen by a feature-selection algorithm.

For this last task, the ClassifierSubSet algorithm offered by Weka was executed. This was configured with the BestFirst algorithm as search method and LinearRegression as base classifier. The leave-one-out method was used for its evaluation. In order to choose features according to their relevance, ten thresholds were defined: 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 95%.

All the linear regression models (meta-learners) were generated using the leave-one-out strategy as evaluation process. The accuracy and the root-mean-square error (RMSE) of each regression model computed were both stored in the experimental database. Later, the same task was repeated but, this time, the MultilayerPerceptron algorithm was used as base classifier. The analysis and discussion of the results are written in Section 4.

Finally, one algorithm recommender was built with the best setting chosen from the analysis of meta-features previously performed. Next, two new data sets were loaded to the system with the aim of uncovering the ranking of algorithms that the recommender offered as outcome and check how good this recommendation was. The results and conclusions drawn are written in the next section.

4. Results and discussion

This section is organised in twofolds: first, the analysis of meta-features is presented and discussed, and next, the feasibility and suitability of meta-learning based recommenders is demonstrated.

4.1. Meta-feature analysis

With the aim of studying and comparing the behaviour of each group of meta-features, two different regressors were used: LinearRegresion and MultilayerPerceptron. Table 2 displays the RMSE obtained in the building of each linear regression model for each algorithm by using all meta-features ("all"), only the complexity ones ("comp"), only the simple ones ("simp"), only the statistical ones ("stat") and, finally, only the landmarkers ("land"). The last column, labelled "Avg. Acc." gathers the average RMSE computed as the mean of the RMSE achieved for each classifier on each data set. This will be used as base value in our experiment. Columns with * denote the RMSE obtained when a feature selection process was applied on all meta-features ("all") or a set of them ("comp*", "stat*", "simp*", "land*") with a threshold of 95%. The value in bold points out the set of meta-features for each classifier that builds the best meta-learner, that means, the one with the lowest RMSE.

The reason why this threshold was chosen can be found in Fig. 3, which shows the average RMSE as result of applying the feature selection process on all meta-features with different thresholds for the eighteen classifiers. As can be observed, the RMSE decreases when the threshold increases, being 95% thus selected. This is a consequence of the fact that, by using Linear Regression, the ClassifierSubSetEval process tends to assign very high values of relevance to most of the meta-features, so that the improvement is barely significative. Suffice it to show that the RMSE is lower than 0.08 with a threshold of 90%, and 0.061 with 95%.

Back to the results shown in Table 2, it can be stated that the feature selection process notably improves the RMSE of all linear regression models

Table 2
RMSE of meta-learners built with LinearRegression for each classifier. The lowest RMSE noted in bold

	all	all*	comp	comp*	land	land*	simp	simp*	stat	stat*	Avg. Acc.
AdaboostM1	0.153	**0.036**	0.091	0.093	0.053	0.048	0.126	0.107	0.143	0.112	0.103
ADTree	0.138	**0.033**	0.087	0.084	0.043	0.040	0.123	0.113	0.114	0.104	0.097
Bagging	0.179	0.043	0.132	0.057	0.043	**0.042**	0.117	0.097	0.118	0.099	0.093
BayesNet	0.254	0.118	0.134	0.116	0.102	**0.100**	0.194	0.181	0.205	0.178	0.154
BFTree	0.181	0.063	0.169	**0.061**	0.065	0.063	0.141	0.118	0.145	0.122	0.114
J48	0.121	**0.038**	0.114	0.063	0.044	0.044	0.122	0.099	0.129	0.103	0.095
Jrip	0.145	0.079	0.179	0.072	0.061	**0.059**	0.137	0.112	0.145	0.114	0.106
LogisticReg	0.071	0.058	0.079	**0.043**	0.054	0.051	0.097	0.086	0.097	0.090	0.084
Multiboost	0.151	0.062	0.129	0.064	0.057	**0.053**	0.133	0.125	0.145	0.125	0.108
MultilayerPerceptron	0.196	0.077	0.218	**0.061**	0.066	0.063	0.131	0.121	0.118	0.103	0.111
Nnge	0.110	0.040	0.075	**0.039**	0.053	0.051	0.112	0.099	0.105	0.097	0.096
OneR	0.140	**0.056**	0.148	0.066	0.071	0.066	0.129	0.117	0.157	0.114	0.110
PART	0.137	**0.044**	0.118	0.092	0.047	0.047	0.106	0.087	0.111	0.094	0.084
RandomForest	0.199	**0.050**	0.137	0.052	0.055	0.054	0.114	0.101	0.122	0.099	0.098
Ridor	0.163	0.053	0.153	**0.049**	0.052	0.051	0.129	0.112	0.146	0.124	0.108
SimpleCart	0.202	**0.088**	0.215	0.125	0.095	0.090	0.170	0.143	0.171	0.146	0.138
SMO	0.228	**0.065**	0.126	0.123	0.106	0.095	0.131	0.112	0.176	0.123	0.120
VotedPerceptron	0.229	**0.092**	0.114	0.107	0.121	0.118	0.162	0.162	0.175	0.141	0.140
Average	0.166	**0.061**	0.134	0.076	0.066	0.063	0.132	0.116	0.140	0.116	0.109
T# best	0	9	0	5	0	4	0	0	0	0	0

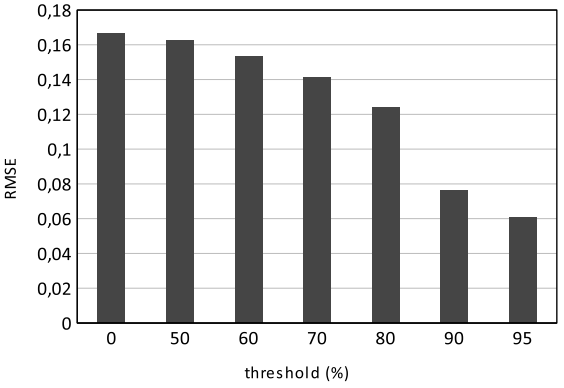

Fig. 3. RMSE of meta-learners built with LinearRegression after applying the feature selection process.

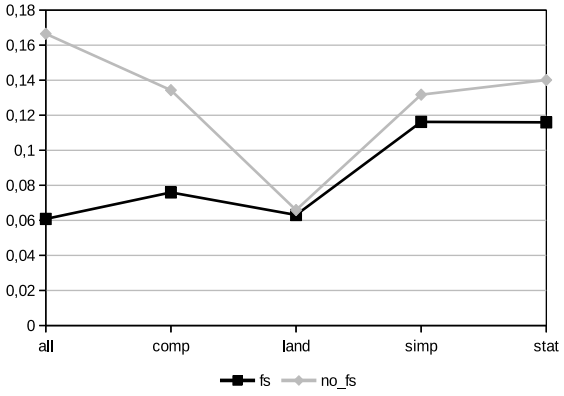

Fig. 4. Comparative of the average RMSE obtained by Linear Regression with (fs) and without (no_fs) feature selection.

built independently of the classifier or the set of meta-features chosen. It is worth noting the improvement in the prediction of the accuracy of classifiers such as VotedPerceptron that reduces the RMSE from 0.229 to 0.092. Moreover, there are some linear regression models which achieve a RMSE lower than 0.05 and even 0.04 by applying feature selection on all the meta-features. That is the case of AdaboostM1, ADtree and J48, with a RMSE of 0.036, 0.033 and 0.038 respectively. Furthermore, nine out of the eighteen algorithms built the best linear regression models when using feature selection on all the meta-features. The same conclusions can be drawn when the results obtained by using feature selection on the complexity meta-features are observed, where the average RMSE decreases from 0.134 to 0.076. In fact, in five times, this setting is the best.

The linear models built from simple and statistical meta-features are quite worse than previous ones with RMSE higher than 0.1 independently of using feature selection or not. However, the behaviour of landmakers must be highlighted since they achieved models with very low RMSE. In fact, the average improvement as result of applying the feature selection process is smaller than in the other cases as can be better visualized in Fig. 4.

Finally, it is worth noting that the average RMSE for predicting the accuracy of an algorithm using meta-learnig is far lower than the RMSE base (see column "Avg. Acc."). This error is higher than 0.1 in eleven out of eighteen algorithms, and higher than 0.08 in the rest, meanwhile this value using all meta-features, or only the complexity ones or only the landmarkers is quite lower (0.061, 0.076 and 0.063 respectively). Moreover, the p value of the two-paired t-test performed to compare these values with respect to the ones in the column "Avg. Acc." is, for all cases, lower than 0.01, so that it can be concluded that the meta-learning process does not only achieve better results, but the difference is very significative.

Next, Table 3 summarises how many times (in percentage) each meta-feature was selected when the feature selection with a threshold of 95% was performed. As can be observed, there are four landmarkers, BestNode, RandomNode, NaïveBayes and

Table 3
Selection of each meta-feature with FS 95% - LinearRegression

Type	Name	% selected
Simple	N# attributes	55.56
	N# instances	33.33
	dimensionality	27.78
Statistical	skewness-avg	16.67
	skewness-min	22.22
	skewness-max	22.22
	kurtosis-avg	16.67
	kurtosis-max	16.67
	kurtosis-min	27.78
Complexity	F1	44.44
	F1v	72.22
	F2	38.89
	F3	22.22
	F4	27.78
	L1	38.89
	L2	27.78
	L3	27.78
	N1	50.00
	N2	33.33
	N3	27.78
	N4	50.00
	T1	22.22
	T2	0.00
Landmarkers	BestNode	50.00
	RandomNode	83.33
	NaiveBayes	50.00
	LinearDiscriminant	33.33
	1-NN	55.56

Table 4
RMSE of meta-learners built with MultilayerPerceptron for each classifier. The lowest RMSE noted in bold

	all	all*	comp	comp*	land	land*	simp	simp*	stat	stat*	Avg. Acc.
AdaBoost	0.058	**0.054**	0.091	0.137	0.076	0.070	0.120	0.111	0.141	0.115	0.103
ADTree	0.095	**0.037**	0.134	0.091	0.071	0.047	0.124	0.107	0.121	0.144	0.097
Bagging	0.062	0.042	0.181	0.105	0.052	**0.040**	0.125	0.099	0.124	0.151	0.093
BayesNet	**0.070**	0.127	0.127	0.136	0.191	0.121	0.176	0.169	0.273	0.184	0.154
BFTree	0.103	**0.076**	0.199	0.153	0.125	0.082	0.157	0.122	0.174	0.150	0.114
J48	0.067	0.053	0.118	0.071	0.057	**0.051**	0.140	0.107	0.128	0.119	0.095
Jrip	0.110	**0.056**	0.237	0.171	0.086	0.074	0.167	0.116	0.122	0.154	0.106
RMSE	0.071	**0.058**	0.124	0.078	0.065	0.065	0.111	0.093	0.138	0.106	0.084
Multiboost	0.079	**0.051**	0.099	0.105	0.081	0.061	0.125	0.125	0.171	0.133	0.108
MultilayerPerceptron	0.144	0.096	0.124	**0.076**	0.086	0.086	0.149	0.115	0.111	0.115	0.111
Nnge	0.098	0.072	0.135	**0.057**	0.078	0.062	0.129	0.101	0.077	0.086	0.096
OneR	0.138	0.060	0.133	0.099	0.074	**0.058**	0.139	0.133	0.170	0.160	0.110
PART	0.062	**0.049**	0.115	0.091	0.075	0.056	0.129	0.087	0.112	0.110	0.084
RandomForest	0.075	**0.052**	0.176	0.067	0.103	0.056	0.128	0.102	0.107	0.109	0.098
Ridor	0.110	0.059	0.154	**0.053**	0.055	0.059	0.134	0.124	0.130	0.143	0.108
SimpleCart	0.170	0.195	0.201	**0.132**	0.170	0.147	0.203	0.143	0.183	0.172	0.138
SMO	0.178	**0.104**	0.163	0.106	0.123	0.116	0.176	0.141	0.164	0.148	0.120
VotedPerceptron	0.185	0.184	0.196	0.144	0.155	**0.108**	0.211	0.145	0.194	0.165	0.140
Average	0.104	0.079	0.150	0.104	0.096	**0.076**	0.147	0.119	0.147	0.137	0.109
T# best	1	9	0	4	0	4	0	0	0	0	0

1-NN, and three complexity measures,F1v, N1 and N4, that were chosen at least a 50% of times, highlighting RandomNode with a 83.33%. This fact explains why landmarkers achieve, alone, a good RMSE in most of the linear regression models, even when the feature selection process is not applied. Nevertheless, simple and statistical meta-features are not, in general terms, highly relevant. Only the number of attributes (N# attributes) shows a selection ratio of 55.56%, being the rest of them below 30% and even 20%.

Thus, these results show that building algorithm recommenders based on linear regressors for predicting the accuracy of each classifier is viable and furthermore highly suitable, since the error ratio is very low. Landmarkers and complexity measures have the highest predictive power, however, when all meta-features are used, included simple and statistical, the best regression models are built for the majority of the classifiers.

Next, the same analysis is performed for meta-learners built with MultilayerPerceptron, a more complex technique than linear regression. Table 4 and Figs. 5 and 6 show the results achieved. As can be observed, the feature selection process, in this case, has a minor effect on the improvement of the RMSE. Moreover, the best results are obtained, on average, with a threshold of 80%. In this scenario, the best average RMSE is achieved by using only landmarkers after applying the feature selection, with a RMSE of 0.076, nearly followed by the average

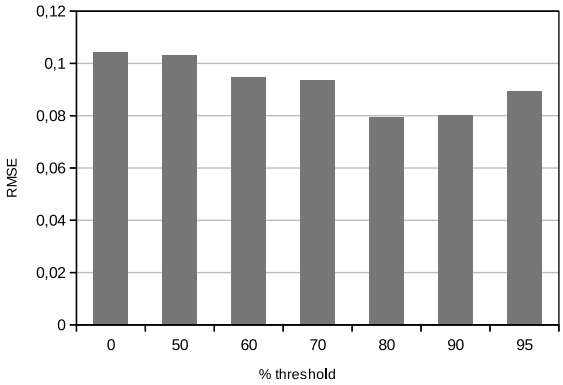

Fig. 5. RMSE of meta-learners built with MultilayerPerceptron after applying the feature selection process.

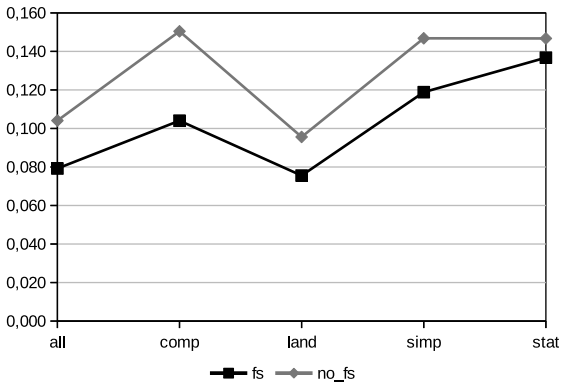

Fig. 6. Comparative of the average RMSE obtained by MultilayerPeceptron with (fs) and without (no_fs) feature selection.

RMSE obtained by using all meta-features, 0.079. Unlike linear regression, complexity measures are not so effective since the models have an average RMSE of 0.104, although, if the results are analysed classifier by classifier, there are four of them, MultilayerPerceptron, NNge, Ridor and SimpleCart, whose results are the best.

Table 5 shows the percentage of times that each meta-feature was selected when a feature selection with a threshold of 80% was performed. These results explain why the feature selection process in this case does not achieve the same degree of improvement that the one obtained when LinearRegression was used and, furthermore, they also explain why the RMSE obtained with MultilayerPerceptron is, in general, higher. As can be observed, the only remarkable meta-feature is a landmark, RandomNode, that was selected 66.67% of times. The other landmarkers and the complexity meta-features are barely selected, a 27.78% and 22.22% of times respectively. In fact, some complexity meta-features, such as F1v, F4 and L3 that were relevant with LinearRegression, are never included in these regression models. Regarding simple and statistical meta-features, these are also barely chosen no more than the 11.11% of times.

Table 5
Selection of each meta-feature with FS 80% - MultilayerPerceptron

Type	Name	% selected
Simple	N# attributes	11.11
	N# instances	0.00
	dimensionality	5.56
Statistical	skewness-avg	11.11
	skewness-min	5.56
	skewness-max	0.00
	kurtosis-avg	0.00
	kurtosis-max	0.00
	kurtosis-min	0.00
Complexity	F1	5.56
	F1v	0.00
	F2	5.56
	F3	5.56
	F4	0.00
	L1	11.11
	L2	22.22
	L3	0.00
	N1	22.22
	N2	11.11
	N3	16.67
	N4	16.67
	T1	5.56
	T2	0.00
Landmarkers	BestNode	5.56
	RandomNode	66.67
	NaiveBayes	22.22
	LinearDiscriminant	27.78
	1-NN	22.22

Table 6
N# of times that LinearRegression (LR) and MultilayerPerceptron (MLP) obtained a lower RMSE after applying a feature selection process

	all*	comp*	land*	simp*	stat*
LR	14	16	15	13	17
MLP	4	2	3	5	1

Finally, Table 6 is shown with the aim of comparing the number of times that LinearRegression and MultilayerPerceptron algorithms obtained a lower RMSE after applying a feature selection process, that means, they built a better model. It is worth noting that LinearRegression achieves a better model in fourteen out of eighteen classifiers when all meta-features are used. This fact is even more remarkable when only complexity meta-features or landmarkers are selected. In the light of these results it can be stated that using a simple technique as LinearRegression for building meta-learners leads to better models than when more complex techniques like MultiplayerPerceptron are applied. In both cases, the feature selection process makes improvements and thus, it should be always performed.

4.2. Predictive power of algorithms recommenders

Next, the predictive power of an algorithm recommender built with LinearRegresion after having applied a feature selection process with a threshold of 95% is demonstrated. For that, two unseen data sets were loaded to the system and their recommendation compared with the best real model built. Columns "P. Acc." and "R. Acc." in Tables 7 and 8 contain the predicted or expected accuracy and the real accuracy obtained for the classifier respectively, and columns "Rank" and "R. Rank" gather their position in the ranking. As can be observed, the recommended classifier for the first data set, SMO, achieves the second highest real accuracy among all classifiers, 85.94%. Moreover, the classifiers with the third, fourth and fifth higher accuracies are ranked in the second, third and fourth position. However, the best real classifier, NNge, is ranked in the fifth place. It is true that the system has not recommended the best classifier, however, it has been able to recommend the second one and rank some of the better classifiers at the top. Furthermore, the worst sixth classifiers are ranked at the lowest places in the list.

The results for the second data set can be observed in Table 8. The first and the second classifier reach

Table 7
Ranking of classifiers for the first unseen data set

Classifier	P. Acc.	R. Acc.	Rank	R. Rank
SMO	84.19	85.94	1	2
Multiboost	83.40	82.81	2	3
LogisticRegression	81.52	79.69	3	4
BayesNet	81.25	78.13	4	5
NNge	79.92	89.06	5	1
MultilayerPerceptron	79.73	85.94	6	2
RandomForest	79.61	78.13	7	5
ADtree	79.21	78.13	8	5
BFtree	77.06	78.13	9	5
SimpleCart	76.78	78.13	10	5
PART	76.37	76.56	11	6
Ridor	76.25	76.56	12	6
Bagging	76.23	73.44	13	8
J48	75.66	75.00	14	7
Jrip	74.60	73.44	15	8
Adaboost	73.94	73.44	16	8
OneR	71.98	73.44	17	8
VotedPerceptron	70.52	45.31	18	9

Table 8
Ranking of classifiers for the second unseen data set

Classifier	P. Acc.	R. Acc.	Rank	R. Rank
RandomForest	94.85	85.92	1	3
Adaboost	90.15	85.92	2	3
Multiboost	88.69	88.73	3	1
Bagging	87.16	88.73	4	1
ADtree	85.67	85.92	5	3
J48	85.54	83.10	6	5
OneR	85.41	88.73	7	1
Ridor	85.34	87.32	8	2
SimpleCart	85.15	83.10	9	5
Jrip	84.86	88.73	10	1
MultilayerPerceptron	84.03	85.92	11	3
Nnge	82.61	85.92	12	3
LogisticRegression	82.37	83.10	13	5
BayesNet	81.98	84.51	14	4
BFTree	81.97	77.46	15	7
SMO	78.40	77.46	16	7
PART	77.06	80.28	17	6
VotedPerceptron	76.30	77.46	18	7

the third higher real accuracy, meanwhile two of the classifiers with the best real accuracy, Bagging and MultiBoost, are classified in the third and fourth position. The reason why Adaboost and RamdonForest are ranked in the two first positions is due to the fact that the error rate in the prediction of the accuracy is higher than in the rest. For example, the predicted accuracy for Multiboost is 88.69% and the real accuracy is 88.73%, having thus a prediction error of only 0.04%, whereas RandomForest is 8.93%. On the other hand, the four worst classifiers, BFTree, PART, SMO and VotedPerceptron are indeed at the bottom of the ranking, so it can be concluded that the meta-learner, as happened previously, works fine on detecting algorithms with low performance, and thus it does not recommend them. Moreover, the difference between the accuracies of the fourth worst classifier, PART, with 80.28% and one of the best classifiers, such as Multiboost with 88.73%, is higher than an 8%, meanwhile the difference between the real accuracy of one of the recommended classifiers, RandomForest, with 85.92%, and Multiboost is lesser than 3%. Hence, it can be stated that the system works suitably ranking those that performed better in the first places and the worst ones in the last positions.

Another important conclusion that this experiment draws is the need of working with a wide variety of algorithms because depending on the meta-features of the data set, an algorithm as SMO can be the best (first data set) or the worst (second data set) choice. This leads to corroborate that if the data mining process for non-expert miners want to be automated with a certain degree of quality, a metalearning-based algorithm recommender is a feasible solution.

5. Conclusions

This paper describes a meta-learning based framework which allows practitioners to discover what algorithm is the most suitable for applying on certain bi-classs data set. Unlike the previous works [6], in this paper, regressors have been used as meta-learners to offer a ranking of algorithms instead of only the best, since most times the difference in accuracy is pretty inappreciable. Likewise, a thorough study about what meta-features are more useful for this end has been carried out on data sets from educational arena. In the light of the results analysed, the landmakers are the most informative meta-features independently of what meta-regressor is used, although the best meta-learners are achieved when all meta-features are utilised and a feature selection process is previously run on them. Another interesting point is that a simpler algorithm such as LinearRegresion behave better as meta-learner than more complex ones, such as MultilayerPerceptron. Finally, the suitability and feasibility of the framework have been tested and very satisfactory results on data sets from the educational arena have been achieved.

In the near future, the framework will be extended to address multi-class problems and nominal attributes where there are still many issues opened [11]. Likewise, other regressors, based on decision trees or rules will be tested and compared with the aim of determining which one is the most suitable for the educational domain. In order to increase the

number of training sets, the artificial generation of data sets will be also assessed.

Acknowledgments

This work has been partially funded by the Spanish Government under grant TIN2014-56158-C4-2-P (M2C2).

References

[1] A. Kalousis and M. Hilario, Model selection via meta-learning: A comparative study, *In Proceedings of the 12th IEEE International Conference on Tools with Artificial Intelligence*, 2000, pp. 406–413.

[2] A.S. Britto Jr, R. Sabourin and L.E.S. Oliveira, Dynamic selection of classifiers-a comprehensive review, *Pattern Recognition* **47**(11) (2014), 3665–3680.

[3] A. Peña Ayala, Review: Educational data mining: A survey and a data mining-based analysis of recent works, *Expert Systems with Applications* **41**(4) (2014), 1432–1462.

[4] B. Pfahringer, H. Bensusan and C. Giraud-Carrier, Metalearning by landmarking various learning algorithms, *In Proceedings of the 17th International Conference on Machine Learning*, Morgan Kaufmann, 2000, pp. 743–750.

[5] C. Köpf, C. Taylor and J. Keller, Meta-analysis: From data characterisation for meta-learning to meta-regression, In *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support*, Meta-Learning and ILP, 2000.

[6] C. Lemke, M. Budka and B. Gabrys, Metalearning: A survey of trends and technologies, *Artificial Intelligence Review* **44**(1) (2013), 117–130.

[7] C. Romero, J.L. Olmo and S. Ventura, A meta-learning approach for recommending a subset of white-box classification algorithms for Moodle datasets, In *Proceedings of the 6th Int. Conference on Educational Data Mining*, 2013, pp. 268–271.

[8] C. Romero and S. Ventura, Educational data mining: A review of the state of the art, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **40**(6) (2010), 601–618.

[9] C. Romero and S. Ventura, Data mining in education, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **3**(1) (2013), 12–27.

[10] D.H. Wolpert, The supervised learning no-free-lunch theorems, In *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, 2001, pp. 25–42.

[11] E. Leyva, A. González and R. Pérez, A set of complexity measures designed for applying meta-learning to instance selection, *IEEE Transactions on Knowledge and Data Engineering* **27**(2) (2015), 354–367.

[12] G. Cavalcanti, T. Ren and B. Vale, Data complexity measures and nearest neighbor classifiers: A practical analysis for metalearning, In *Proceedings of the IEEE 24th International Conference on Tools with Artificial Intelligence*, 2012, pp. 1065–1069.

[13] J.L. Olmo, C. Romero, E. Gibaja and S. Ventura, Improving meta-learning for algorithm selection by using multi-label classification: A case of study with educational data sets, *Int J Computational Intelligence Systems* **8**(6) (2015), 1144–1164.

[14] J. Luengo and F. Herrera, An automatic extraction method of the domains of competence for learning classifiers using data complexity measures, *Knowledge Information Systems* **42**(1) (2015), 147–180.

[15] J. Rice, The algorithm selection problem, *Advances in Computers* **15** (1976), 65–118.

[16] J. Vanschoren, H. Blockeel, B. Pfahringer and G. Holmes, Experiment databases, *Machine Learning* **87**(2) (2012), 127–158.

[17] J.W. Lee and C. Giraud-Carrier, Automatic selection of classification learning algorithms for data mining practitioners, *Intelligent Data Analysis* **17**(4) (2013), 665–678.

[18] M.E. Zorrilla and D. García-Saiz, Meta-learning: Can it be suitable to automatise the KDD process for the educational domain? In *Proceedings of the Second International Conference on Rough Sets and Intelligent Systems Paradigms*, 2014, pp. 285–292.

[19] M.E. Zorrilla and D. García-Saiz, Meta-learning based framework for helping non-expert miners to choice a suitable classification algorithm: An application for the educational field, In *Proceedings of the 7th International Conference on Computational Collective Intelligence*, 2015, pp. 431–440.

[20] M. Hilario and A. Kalousis, Building algorithm profiles for prior model selection in knowledge discovery systems, *Engineering Intelligent Systems* **8** (2002), 180–183.

[21] M.M. Molina, J.M. Luna, C. Romero and S. Ventura, Metalearning approach for automatic parameter tuning: A case study with educational datasets, In *Proceedings of the 5th International Conference on Educational Data Mining*, 2012, pp. 956–961.

[22] M. Reif, A. Leveringhaus, F. Shafait and A. Dengel, Predicting classifier combinations, In *Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods*, 2013.

[23] M. Reif, F. Shafait, M. Goldstein, T. Breuel and A. Dengel, Automatic classifier selection for non-experts, *Pattern Analysis and Applications* **17**(1) (2014), 83–96.

[24] P. Kordík and J. Cerný, On performance of meta-learning templates on different datasets, In *Proocedings of the IEEE World Congress on Computational Intelligence*, 2012, pp. 1–7.

[25] R. Vilalta and Y. Drissi, A perspective view and survey of meta-learning, *Artificial Intelligence Review* **18** (2002), 77–95.

[26] S. Newell and M. Marabelli, Strategic opportunities (and challenges) of algorithmic decision-making: A call for action on the long-term societal effects of 'datification', *The Journal of Strategic Information Systems* **4** (2015), 3–14.

[27] S. Segrera, J. Pinho and M.N. Moreno, Information-theoretic measures for meta-learning, In *Proceedings of the 3rd international workshop on Hybrid Artificial Intelligence Systems*, 2008, pp. 458–465.

[28] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, The kdd process for extracting useful knowledge from volumes of data, *Commun ACM* **39**(11) (1996), 27–34.

[29] T.K. Ho, Geometrical complexity of classification problems. CoRR, cs. CV/0402020, 2004.

[30] Y. Peng, P. Flach, C. Soares and P. Brazdil. Improved dataset characterisation for meta-learning. In S. Lange, K. Satoh and C. Smith, editors, Discovery Science, volume 2534 of Lecture Notes of Computer Science. Springer Berlin/Heidelberg, 2002, pp. 193–208.