Selecting Machine Learning Algorithms using Regression Models

Tri Doan
Department of Computer Science
University of Colorado
Colorado Springs
tdoan@uccs.edu

Jugal Kalita
Department of Computer Science
University of Colorado
Colorado Springs
jkalita@uccs.edu

***Abstract*—In performing data mining, a common task is to search for the most appropriate algorithm(s) to retrieve important information from data. With an increasing number of available data mining techniques, it may be impractical to experiment with many techniques on a specific dataset of interest to find the best algorithm(s). In this paper, we demonstrate the suitability of tree-based multi-variable linear regression in predicting algorithm performance. We take into account prior machine learning experience to construct meta-knowledge for supervised learning. The idea is to use summary knowledge about datasets along with past performance of algorithms on these datasets to build this meta-knowledge. We augment pure statistical summaries with descriptive features and a misclassification cost, and discover that transformed datasets obtained by reducing a high dimensional feature space to a smaller dimension still retain significant characteristic knowledge necessary to predict algorithm performance. Our approach works well for both numerical and nominal data obtained from real world environments.**

***Keywords*-Meta-learning; regression; dimensionality reduction; combined metric**

I. Introduction

Learning from data is of interest to many disparate fields such as banking, bioinformatics, business, computer vision and education. The field of data mining uses a large collection of machine learning algorithms whose goal is to extract useful information from collected data. For any given dataset, a common question is which learning algorithm is best suited for it. Performing experiments with several algorithms using the data, or getting advice from machine learning experts, can help assess which algorithms might be the best candidates, but this is not always practical.

Using the idea of meta-learning, we solve the problem of selecting a machine learning algorithm for a particular dataset by supervised learning. In this work, we represent the efficient way to deal with a non-standard format in real-world dataset to obtain training data. Our work is further address the problem of well-known algorithm selection recently.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 motivates the solution, followed by our approach in Section 4. Section 5 describes our experiments. Discussion is in Section 6. Finally, Section 7 summarizes the paper and provides directions for future study.

II. Related Work

In the past, researchers have attempted to predict algorithm behavior based on data characteristics.

Example approaches include STATLOG [1], and METAL [2] that use machine learning on acquired knowledge of how various machine learning algorithms perform on various datasets. To construct meta-knowledge, statistical summaries and non-statistical information (such as number of instances and number of classes) are used. Metrics from information theory (such as entropy and mutual information) are used for nominal data. An approach called landmarking [3] captures a set of informative features computed from the selected learners (e.g., C5.0tree) on selected datasets. Going further, [4] introduce the use of model-based properties in meta-learning. These features include characteristics of induced decision trees (e.g., nodes per attribute, nodes per instance, or average gain-ratio difference) using the same datasets [5].

We have identified several issues with past approaches to select machine learning algorithms. For example, statistical features that have been used in past studies [1], [6], [7] include the mean of summary statistics (e.g., means of standard deviations, skewness, and kurtosis across all features) of a dataset. We believe that averaging values of summary statistics dilutes statistical meaning. For instance, a left-skew on one attribute might cancel out the right-skew on another attribute in the final mean of skewnesses. The mean of skewness or kurtosis values across all features loses its discriminating character when the distribution is non-uniform [6].

Using many or all features may also be a source of problem for both real datasets and meta-knowledge derived from them. A smaller optimized number of meta-features may improve the quality of training data to produce a better predictive meta-model. Researchers deal with a high number of dimensions in two ways: feature selection and feature extraction. Feature selection retains the most discriminating features [8]. Feature extraction, on the other hand, generates a new set of features as predictors in the form of composite attributes [9].

As a breakthrough of among studies of algorithm selection, *Auto-Weka* [10] allows one to find the best algorithm with corresponding optimal parameters among algorithm of interest. While its result provides optimal parameters for each algorithm, it suffers a computational problem that our work desire to address. Two main reasons include non heuristic to select algorithms to perform the hyper-parameter search for a given dataset and each execution starting from scratch. First reason results in wasting time for searching hyper-parameter of algorithms that are actual low performance in a specific data domain. Second reason lies on the approach such that Auto-Weka is built ignores knowledge gained from past experience.

In addition, there is very little work on algorithm selection using regression, especially a regression tree model. Our work study the use of the family of tree models and demonstrate the advantage of tree model.

III. Meta-learning in steps

- Create a training set from original datasets and evaluate performance of algorithm on these transformed datasets.
- Select a best regression model and predict algorithm performance on unknown dataset.
- Generate a ranked list of machine learning algorithms.

To create meta-data, each original dataset is used to generate one meta-instance in the training set. Since real-world datasets come with different numbers of features, we transform each original dataset into a dataset with a fixed number of features before performing training. Our approach uses dimension reduction to generate a fixed number of features. Transformed datasets can be generated with a feature generation approach which takes all features from original dataset into account. In addition, the runtime for the transformed dataset is likely to be lower than for the original datasets because computational time for a large number of features often results in higher run time of an algorithm. As each feature has its own predictive power, a feature selection approach may not have provided an adequate solution to generate the same fixed number of features. The reason is that if we want the features to guarantee high predictive power, feature selection is likely to select different numbers of features for different datasets whereas our meta-learning training set requires a fixed number of features.

With transformed datasets obtained, we use supervised learning to solve the problem of predicting performance of specific algorithms on specific datasets. We apply both linear regression and non-linear regression. Finally, the outcomes of the regression methods are used to generate a ranked list of algorithms by performance for a specific dataset. Intuitively, using regression is suitable as data mining practitioners may be interested in comparing how various algorithms perform on a dataset first by performing regression to predict performance of individual algorithms on the dataset.

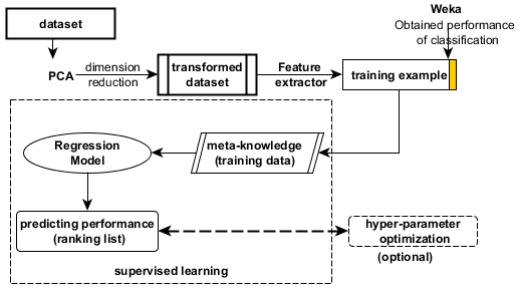

Figure 1. Proposed model for algorithm selection

Table I
META-FEATURES USED

Feature	Description
Algorithm	Data mining algorithm's name
ClassInt	Ratio of no of classes to instances
AttrClass	Ratio of no of features to no of classes
Median*	Middle value in a dataset
Mean*	Average value in a dataset
Std*	Standard deviation
Skewness*	Measure of asymmetry of the prob. dis.
Entro-Class	Class entropy for target attribute
TotalCorr	Amount of information shared among variables
Performance	Measure of performance of each algorithm

* 4 statistics computed for each of the four meta-features

Table II
DATA MINING ALGORITHMS IN THIS STUDY

Learner	Description
J48	Non-commercial decision tree C4.5
Decision Stump	A decision stump
PART	A version of C4.5 using decision list
Decision Table	Simple decision table
JRip	Propositional rule learner
OneR	Uses minimum-error attribute
ZeroR	0-R classifier
IBk	K-NN classifier
KStar	Entropy-based
LWL	Locally Weighed Learning
Naive Bayes	Naive Bayes classifier
AdaBoost M1	Boosting a nominal class classifier
Bagging	Reduces variance
Stacking	Combines the output from others
Logit Boost	Additive logistic regression classifier
Random Committe	Ensemble of randomizable base classifiers
Random Forest	Forest of random trees
Vote	Uses majority vote to label new instance
Logistic	Logistic regression with a ridge estimator
Multilayer Perceptron	Neural network with backpropagation
Simple Logistic	Linear logistic regression
SMO	Sequential minimal optimization
SVM	Support Vector Machine

IV. Approach

Figure 1 illustrates our proposed approach. At first, the original datasets are preprocessed to generate training examples. To do this, each dataset is transformed into a corresponding intermediate dataset with a fixed number of features. This allows us to generate meta-features for the training set. Weka is used to obtain actual classification

performance of all classifiers using the transformed datasets. These values represent the labels in training examples corresponding to generated features for each dataset. Next, we construct our regression model using the set of training examples. Finally, we use this model to produce predicted performance and the ranked list. The whole process is described in following sub-sections.

A. Dimensionality Reduction with variable feature issue

Dimensionality reduction uses different techniques to generate a new small set of features from an original dataset. A commonly well-known technique for dimension reduction is Principal Components Analysis (PCA). We use PCA due to its computational efficiency, say compared to the Single Value Decomposition method [11]. To work with PCA, a nominal feature needs to be converted into numeric. We take a straightforward approach: A nominal attribute that takes *m* distinct values, is converted to *m* new binary features. As non-linear correlation cannot be detected by PCA, we adapt the approach by [12] to include mutual information by discretizing all numeric values.

Table III
AVERAGE ERROR RATE WITH NUMBER OF FEATURES

No of Features	**2**	**3**	**4**	**5**	**6**
Avg Error Rate	0.3873	0.3859	**0.3701**	0.3705	0.3701

One important question is how many features to generate in the transformed dataset for a typical classification problem. To justify a certain number of features, we perform experiments by reducing the dimension of the datasets into 2, 3, 4, 5 and 6 features. For each dataset, we implement classification for each of the datasets after reducing dimensionality using 23 different algorithms and record the accuracy measurement in each classification problem. Accuracy metric, which is correlated with the SAR metric used later in the paper (explained in Section 4.3), is used in this preliminary experiment to justify the choice of the number of features for simplicity because it is generated directly from most classifier algorithms. The error rate (determined by 1-accuracy) of each transformed dataset is used to compute the averages of error rate for each number of features (see Table III). The number of features with the lowest average error rate is selected as the number of features used for dimensionality reduction. Table III indicates that the more the number of features, the lower the average error rates generated in classification tasks. However, it is only true in case of existing independent features. Our experiment with up to 10 generated features (not present in paper) confirm similar pattern. With current features generated, the feasible choices are 4, 5 and 6. It is reasonable not to use higher numbers (e.g., from 7 onward) as dimensionality reduction techniques cannot generate more features than the number of features in original datasets. Choosing 5 or 6 features for transformed data is feasible but if we do so, we will exclude original datasets with fewer than 5 or 6 features. We choose 4 as the number of features to produce transformed datasets to generate meta-data (training set) for our study.

B. Meta-features

Table I describes the meta-features of the training set which are obtained from transformed dataset (see Figure 1). The first feature indicates the learner under consideration whereas the next two features describe the relation between the number of classes to the number of attributes and the number of instances, respectively. The set of four summary statistics, viz., Median, Mean, Standard deviation and Skewness, is computed for each of the new mixed features of the compressed datasets. It results in 4*4 =16 meta-features based on statistical summaries. These includes Median1, Median2, Median3, Median4, Mean1, Mean2, Mean3, Mean4, Std1, Std2, Std3, Std4, Skewness1, Skewness2, Skewness3, Skewness4 features. These four statistics provide good predictive power while keeping the number of features in the meta-data relatively low. We also use two information theory features such as class entropy for target attribute and total correlation. The last attribute represents the learner's actual performance, which is an experimentally derived value. This attribute (*Performance*) indicates the performance of each algorithm on a particular dataset based on the SAR metric (**S**quared error, **A**ccuracy, **R**OC Area), described later.

Among these information theoretic metrics, class entropy (the *EntroClass* attribute) indicates how much information is needed to specify one target class whereas total correlation (the *TotalCorr* attribute) measures the amount of information shared among the attributes. To offset the possible loss of critical information that might be caused by inappropriate bin boundaries in discretization, we include an additional feature: the ratio of the number of classes to the dataset's size (the *ClassInt* attribute) to measure the trade-off between the number of instances and the number of target classes. We also calculate the ratio of the number of features to the number of target classes (the *AttrClass* attribute).

C. Measurement metrics

In reality, different evaluation metrics can present conflicting results when assessing algorithms' performance, especially when multiple algorithms based on different data mining approaches are involved. This problem has been discussed extensively in the study by [13], [14]. Using common accuracy metric for algorithm performance has some drawbacks. It does not allow to show the different importance of performance on each class. Accuracy is also known to be sensitive to unbalanced classes when the assumption of a normal distribution between classes is not guarantee. As a result, to evaluate an algorithm's performance, we propose to use a combined metric that takes

advantage of three commonly used metrics in classification. This metric, SAR proposed in the study [13], is computed as SAR = [Accuracy+AUC+(1-RMSE)]/3 where AUC and RMSE are Area Under the Curve, and Root Mean Square Error, respectively. SAR $\in$ [0,1] where the higher the better.

On the other hand, we use RMSE metric for regression task when we provide the comparison of several candidate models for predicting algorithm performance. In regression, RMSE is more suitable as it indicates the difference between observed and predicted values.

To generate a list of algorithms for classifying a particular dataset, we use a minimum performance threshold (0.6) using the SAR metric. With a threshold parameter value bigger than 0.5, the final list of algorithms includes only those with high performance measured by the combined metric. Using threshold of 0.6 can be justified since including less performance algorithms in the final list is not productivity. Retaining only high performed algorithms for a specific dataset reduces significantly computational expense when we decide either more features need to be collected in original dataset or further hyper-parameters is required. Finally, we generate a ranked list of algorithms by predicted performance indicating how a particular algorithm may behave given an unknown dataset.

D. Data source

We use two types of datasets, real and synthetic. From the UCI repository [15], we select 73 datasets from a variety of domains. The collection of data includes many datasets that have been used in similar work on algorithm selection [16], [17], [10].

As real world datasets often come with noise and imprecision due to error in measuring devices and human error, drawing precise conclusions from mined results suffers from the fact that any assumption regarding data distribution cannot be guaranteed. The artificial datasets also counter the limitation that selected real datasets cover only a small number of data domains. We use generators from the open source tool *scikit-learn* [18] to generate synthetic datasets with at least four attributes. The number of classes vary from 2 to 10 with a randomized number of instances between 100 and 4,000. Random noise is introduced in all of these datasets. From generated synthetic datasets, we select only datasets with at least 4 features, resulting in 37 synthetic datasets in a collection of 100 datasets.

E. Regression Models

Our proposed predicting model generates performance for each of algorithms of choice. These predicted performances are used to compute a final ranked list. Among several regression models, we use Regression Tree models and other state-of-the-art models. The basic idea of regression tree is to split data space into many subregions so that a model tree can obtain high accuracy by computing the regression model with small sets of predictors separately in these sub-regions. This feature gives the Regression Tree the ability to apply linear models to non-linear data since non-linear data is common seen and is also in our case. They are Classification and Regression Trees (CART) [19], Conditional Tree [20], Model Tree [21], Rule Based System [22], Bagging CART [23], Random Forest [24], Gradient Boost Regression [25] and Cubist [26].

- The CART tree splits attributes to achieve minimize a loss function. Each final split determines a sub-region of data space that indicate a linear relationship.
- The Conditional Decision Tree (CDT) applies statistical tests to select split points of attributesto avoid selection bias with splits.
- The Model Tree represents each leaf as a linear regression model. Model tree aims at use reduction of error rate at each node when constructing a tree.
- The Rule based system simplifies a Decision Tree by removing parts of rules having low predictive power to avoid overfitting.
- Bagging CART uses bootstrapping with aggregation regression to reduce the variance of prediction. Each model can be built independently.
- Random Forest uses a random selection of features to split each node with bootstrap samples when building trees.
- Gradient Boost Regression extend the AdaBoost [27] using gradient boosting. It adds new models to learn misclassification errors in order to reduce bias.
- The Cubist tries to reduce the condition or a rule without increasing error rate. Cubist can adjust the model prediction using a training set to improve its performance.

The remaining models include Neural Network [28], Support Vector Regression [29], K-Nearest Neighbor [30], Partial Least Squared or PLS [31], Ridge Regression [32], Least Angle Regression or LARS [33], Elastic Net [34] and Multi Variate Adaptive Regression Splines or MARS [35].

- Neural Network connects its predictors to response through its hidden units. Each unit receives information from previous layers and generates output to next layer.
- Support Vector Regression (SVR) searches data points to support a regression. SVR uses a loss function with penalty for high performance in the presence of outliers.
- K-Nearest Neighbor Regression (KNN regression) locates K nearest neighbors in the predictor space to predict new value using the summary statistic
- The PLS model tries to obtain a linear combination of independent variables that can maximize the covariance needed to separate groups.
- The Ridge uses regularization to penalize by shrinking the coefficients toward 0 to counter against highly correlated predictors in a dataset.

Table IV
DATASETS USED AS EXAMPLES TO DEMONSTRATE THE FINAL RANKING LIST

Name	Nominal	Numeric	Class	Instances
Real data				
credit-g	13	7	2	1000
Kr-vs-kp	37	0	2	3196
Abalone	1	7	28	4117
Waveform	0	40	3	5000
Shuttle	9	0	7	58000
Synthetic data				
art-02	0	19	5	1038
art-07	0	23	3	4430
art-12	0	12	10	1276
art-17	0	21	10	3266
art-24	0	18	2	3552

- LARS is based on the LASSO model [36]. It calculate its move in the least angle direction for the next step among the currently most correlated covariates.
- The Elastic Net is a generalized combination of two penalized regression techniques: L1 (LASSO) and L2 (Ridge) introduced by [32] to exploit both advantages.
- MARS initially use surrogates features with only one or two predictors at a time that indicate clearly linear relationship to produce the best fit given the initial set.

Our approach differs from [37] and [38], which either use only artificial datasets (the former) or associate a set of rules with learner's performance (the latter).

F. Evaluation approach

We use two statistical significance tests to assess the result of our approach: Spearman's rank correlation test [39] to measure how close the two ranks are, other is Friedman test to validate true ranking which is robust in case of normal distribution is not guarantee.

V. EXPERIMENTS

Experiments are conducted on a system with Intel Core i5, CPU 2.4Ghz and 6GB RAM. Weka API is used for all classification tasks, Python scripts for creating synthetic datasets and R scripts for remain works.

We use the stratified k-fold method [40] for tackling real world unbalanced datasets We estimate the predicted performance using the regression models for all algorithms (refer to Table II). Finally, predicted and observed values of performance of algorithms based on the test set are ordered and the following statistical tests are performed.

Using Spearman's rank correlation test report in Table VII, we see that all Spearman coefficients fall into four groups (refer Table V): very strong (such as Neural Net, MARS, Bagging CART, Cubist, Conditional Decision Tree, Random Forest), strong (SVR, CART, Model Tree, Rule based System) and moderate (KNN, PLS, Ridge Regresion, LARS and Elastic Net) and weak (Gradient Boost). However, all p-values are significantly small, which imply strong evidence against the null hypothesis or that we reject the null hypothesis. On the other hand, Friedman rank sum test comes out as: χ^2 = 186.2428, df = 8, p-value = 2.2e-18. Based on the result, we reject the null hypothesis that all algorithms are the same at the 0.005 level. All codes in the study are provided in https://github.com/uccs-tdoan/algorithmSelection

Performances below and above the threshold (.6) with the horizontal line indicates performance worse or better than random guess, respectively (see Figures 2 and 3). These diagrams illustrate how algorithms are expected to perform in case of unknown datasets (in these figures, we have 5 real world and 5 synthetic datasets).

VI. DISCUSSION

Given training data obtained, we demonstrate the experiment to select the best regression model that we use to generate predicted performances. The RMSE values in Table VI show that all the regression models are competitive in prediction of algorithm performances. While the RMSEs are less than 1 for all models, Cubist stands out as the best model in this study. Especially, the largest RMSE produced by Model Tree (Gradient Boost model) is still good compared to the smallest RMSE produced by the best model (MARS) under the category of other linear and non-linear regression models. In reality, MARS is considered to be among the most appealing models in regression. LARS, Elastic Net, PLS, Neural Net perform well following the MARS model. Among the tree models, Model Tree and Gradient Boost are considered comparable to the others.

Table VII shows a monotonically increasing relationship between predicted ranks and observed ranks (indicated by ρ >0 Spearman's rank coefficient). This positive rank correlation implies the validity of generated ranks where the predicted high rank is consistent with the observed rank. These results support our rank list based on predicted performance. Figure 2 and Figure 3 illustrate the predicted performance with error ranges for all 23 algorithms using examples generated by the Cubist model on few extra real world and synthetic datasets, respectively. When the lowest error is higher than the threshold (dotted line), we are confident that the corresponding algorithm is capable of working well for the dataset and we include it in the ranked list. The predicted values with error ranges imply further improvement may be possible using hyperparameter optimal search if tuning option is available.

Table V
SPEARMAN RANK COEFFICIENT

Coefficient	very weak	weak	moderate	strong	very strong
range	.001-.19	.2-.39	.4-.59	.6-.79	.8-1

The top 5 rank lists indicate that no single learner dominates on all datasets (Table VIII). The well-known

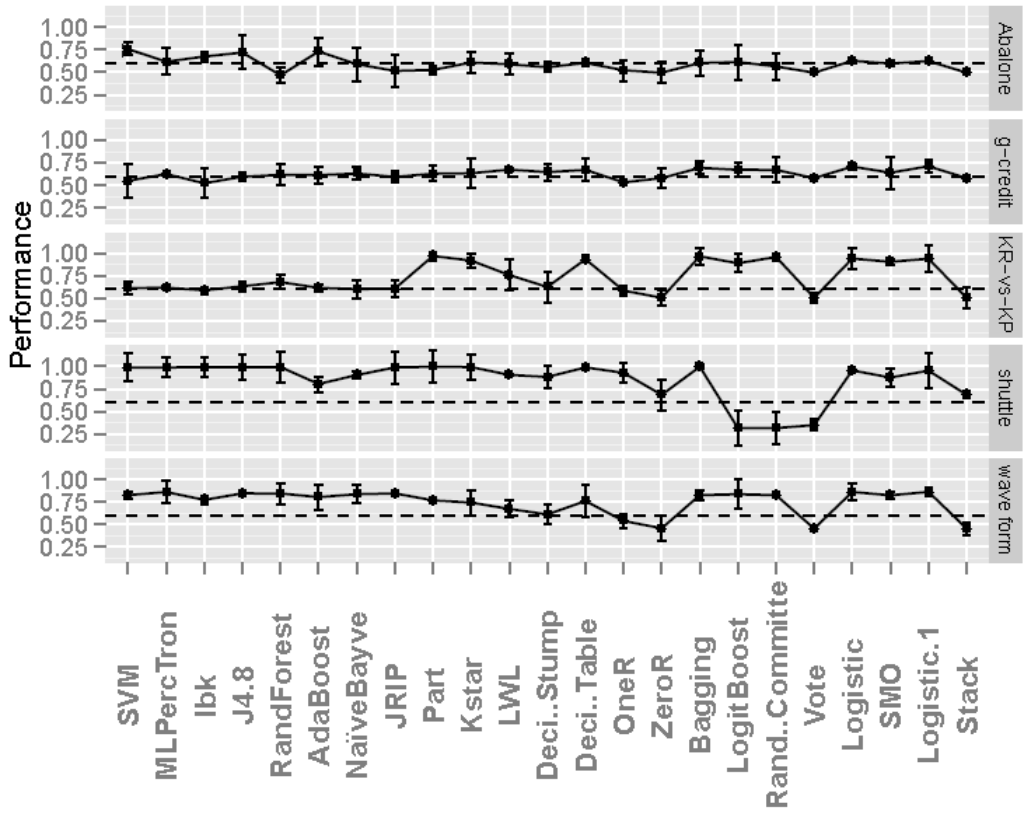

Figure 2. Performance of algorithms on realworld examples.

Table VI
RMSEs BY MULTIPLE REGRESSION MODELS

Tree Models	RMSE	**Other Models**	RMSE
CART	0.9291	**Neuron Net**	0.9699
Conditional D.T.	0.9166	**SVR**	0.9714
Model Tree	0.9308	**KNN**	0.9692
Rule Based System	0.9166	**PLS**	0.9669
Bagging CART	0.9251	**Ridge**	0.5731
Random Forest	0.9216	**LARS**	0.9668
Gradient Boost	0.9439	**Elastic Net**	0.9668
Cubist	**0.9025**	**MARS**	0.9626

Table VII
SPEARMAN 'S RANKING CORRELATION

Tree Models	Spearman rank coef.	**Other Models**	Spearman rank coef.
CART	0.7721	**Neuron Network**	0.8104
Conditional D.T.	0.8856	**SVR**	0.7246
Model Tree	0.6438	**KNN**	0.4849
Rule Based Sys.	0.6438	**PLS**	0.5619
Bagging CART	0.8374	**Ridge**	0.5731
Random Forest	0.8977	**LARS**	0.5790
Gradient Boost	0.3470	**Elastic Net**	0.5790
Cubist	0.8807	**MARS**	0.8385

SVM learner appears in half of all top 5 rank lists but is recommended as first choice only one time across the real world and synthetic datasets. Some learners such as J48 (decision tree) and Part often occur in the top 5 lists and thus can be use as baseline when algorithms are compared or new learning algorithms are developed. Several learners such as KStar, LWL, Logistic with long training time also make their way into the ranking lists. In Figure 2, we observe that only half the algorithms are "suitable" or the *abalone* dataset whereas the *shuttle* dataset has a wide range of "suitable" algorithms. Half the "suitable" algorithms have high performance on the second dataset, implying choices

Table VIII
RANKING OF ALGORITHM PERFORMANCE ON DATASETS

Rank	First	Second	Third	Fourth	Fifth
abalone	SVM	AdaBoost	J48	IBk	Logistic
waveform	**S.Logistic**	Logistic	MultiL	J48	JRip
shuttle	**Part**	Bagging	IBk	Kstar	RForest
g-credit	**SLogistic**	Logistic	Bagging	LWL	LBoost
KRvsKP	Part	Bagging	RandC.	Logistic	SLogistic
art-02	MultiL	SVM	IBk	Logistic	simpleL
art-07	MultiL	IBk	RandomC	Logistic	RForest
art-12	SVM	IBk	MultiL	RandomC	SMO
art-17	IBk	MultiL	SVM	SMO	RandC
art-24	MultiL	SVM	IBk	Logistic	RandC

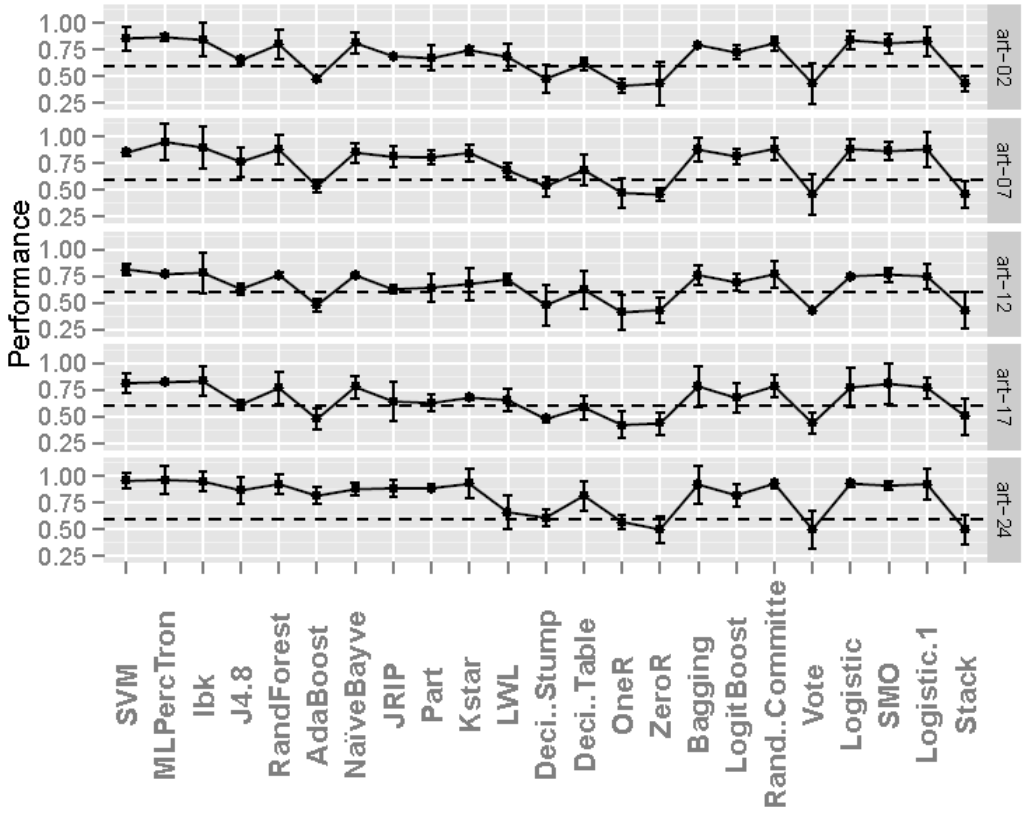

Figure 3. Performance of algorithms on synthetic datasets.

for practitioners.

We note that our rank results obtained with the combined SAR metric are more robust than those obtained only with the accuracy metric [7]. For instance, [7] rank MultiLayer Perceptron in first place but we rank it seventh due to low AUC of this algorithm's performance on *abalone*.

VII. Conclusion and Future Work

Given an unknown dataset, our proposed approach are able give a robust ranked list of learning algorithms with default parameter setting. Furthermore the ranked list can be use to select only feasible algorithms for Auto-Weka execution. One limitation in this work is the minimum 4 features of original datasets. We will address this limitation in future work and investigate the practical use of our proposed approach in a field of NLP (Natural Language Processing). There is a possible extension of this work including integration with Auto-Weka and combination of performance with run time factor. Our future work also is to extend this study in case of a large scale data, particular for big data challenge.

References

[1] R. King and C. e. a. Feng, "Statlog: comparison of classification algorithms on large real-world problems," *Appl Artif Intell an Intern Journal*, vol. 9, no. 3, pp. 289–333, 1995.

[2] H. Berrer and I. e. a. Paterson, "Evaluation of machine-learning algorithm ranking advisors," in *In Proceedings of the PKDD-2000 Workshop on DataMining, Decision Support, Meta-Learning and ILP*. Citeseer, 2000.

[3] H. Bensusan and C. Giraud-Carrier, "Discovering task neighbourhoods through landmark learning performances," in *Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 325–330.

[4] Y. Peng and P. A. e. a. Flach, "Decision tree-based data characterization for meta-learning," 2002.

[5] H. Bensusan and C. G. e. a. Giraud-Carrier, "A higher-order approach to meta-learning." in *ILP Work-in-progress reports*, 2000.

[6] C. Castiello and G. e. a. Castellano, "Meta-data: Characterization of input features for meta-learning," in *Modeling Decisions for Artificial Intelligence*. Springer, 2005, pp. 457–468.

[7] S. Ali and K. A. Smith, "On learning algorithm selection for classification," *Appl Soft Compu*, vol. 6, no. 2, pp. 119–138, 2006.

[8] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[9] H. Liu and H. Motoda, *Feature extraction, construction and selection: A data mining perspective*. Springer, 1998.

[10] C. Thornton and F. e. a. Hutter, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD*. ACM, 2013, pp. 847–855.

[11] L. De Lathauwer and B. e. a. De Moor, "Singular value decomposition," in *Proc. EUSIPCO-94, Edinburgh, Scotland, UK*, vol. 1, 1994, pp. 175–178.

[12] D. N. Reshef and Y. A. e. a. Reshef, "Detecting novel associations in large data sets," *science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[13] R. Caruana and A. Niculescu-Mizil, "Data mining in metric space: an empirical analysis of supervised learning performance criteria," in *Proceedings of the 10th ACM SIGKDD*. ACM, 2004, pp. 69–78.

[14] C. Ferri and J. e. a. Hernández-Orallo, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.

[15] C. Blake and C. J. Merz, "{UCI} repository of machine learning databases," 1998.

[16] R. Leite and P. e. a. Brazdil, "Selecting classification algorithms with active testing," in *ML and Data Mining in Pattern Recognition*. Springer, 2012, pp. 117–131.

[17] M. Reif and F. e. a. Shafait, "Meta-learning for evolutionary parameter optimization of classifiers," *Machine learning*, vol. 87, no. 3, pp. 357–380, 2012.

[18] F. Pedregosa and G. e. a. Varoquaux, "Scikit-learn: Machine learning in Python," vol. 12, pp. 2825–2830, 2011.

[19] L. Breiman and J. e. a. Friedman, *Classification and regression trees*. CRC press, 1984.

[20] T. Hothorn, K. Hornik, and A. Zeileis, "Unbiased recursive partitioning: A conditional inference framework," *Journal of Computational and Graphical statistics*, vol. 15, no. 3, pp. 651–674, 2006.

[21] J. R. Quinlan *et al.*, "Learning with continuous classes," in *5th Australian joint conference on artificial intelligence*, vol. 92. Singapore, 1992, pp. 343–348.

[22] G. Holmes, M. Hall, and E. Prank, *Generating rule sets from model trees*. Springer, 1999.

[23] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[24] ——, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[25] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[26] J. R. Quinlan, "Combining instance-based and model-based learning," in *Proceedings of the Tenth International Conference on Machine Learning*, 1993, pp. 236–243.

[27] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*. Springer, 1995, pp. 23–37.

[28] C. M. Bishop *et al.*, "Neural networks for pattern recognition," 1995.

[29] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.

[30] O. Kramer, *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Springer, 2013.

[31] H. Wold *et al.*, "Estimation of principal components and related models by iterative least squares," *Multivariate analysis*, vol. 1, pp. 391–420, 1966.

[32] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[33] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[34] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[35] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, pp. 1–67, 1991.

[36] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, pp. 267–288, 1996.

[37] C. Köpf and C. e. a. Taylor, "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*. Citeseer, 2000.

[38] H. Bensusan and A. Kalousis, "Estimating the predictive accuracy of a classifier," in *Machine Learning: ECML 2001*. Springer, 2001, pp. 25–36.

[39] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer, 2013.

[40] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," in *Encyclopedia of database systems*. Springer, 2009, pp. 532–538.