EVALUATING PREDICTION STRATEGIES IN AN ENHANCED META-LEARNING FRAMEWORK

Silviu Cacoveanu, Camelia Vidrighin and Rodica Potolea
Technical University of Cluj-Napoca, Cluj-Napoca, Romania

Keywords: Meta-learning Framework, Data Set Features, Performance Metrics, Prediction Strategies.

Abstract: Finding the best learning strategy for a new domain/problem can prove to be an expensive and time-consuming process even for the experienced analysts. This paper presents several enhancements to a meta-learning framework we have previously designed and implemented. Its main goal is to automatically identify the most reliable learning schemes for a particular problem, based on the knowledge acquired about existing data sets, while minimizing the work done by the user but still offering flexibility. The main enhancements proposed here refer to the addition of several classifier performance metrics, including two original metrics, for widening the evaluation criteria, the addition of several new benchmark data sets for improving the outcome of the neighbor estimation step, and the integration of complex prediction strategies. Systematic evaluations have been performed to validate the new context of the framework. The analysis of the results revealed new research perspectives in the meta-learning area.

1 INTRODUCTION

Ever since the beginning of the information age, companies, organizations, universities all around the world have been taking advantage of technological advances to store large amounts of data. The increase of virtual space made available by new and cheaper storage devices has encouraged people to keep records specific to all activities taking place in their institutions. Advances in database systems have made searching and organizing the stored data easier for experts – but the technology for automatically obtaining valuable information from this data has only recently started to gain popularity. Data mining approaches employ methods from different fields, such as statistics, artificial intelligence, meta-learning, to induce models from large amounts of data. These models enable the characterization of data by summarizing the class under study in general terms, discovering frequently occurring subsequences (in transactional datasets), classifying new data, predicting numeric values, grouping similar items in a database, analyzing outliers for fraud detection and analyzing the trends of objects whose behavior changes over time.

An important step in the data mining process is selecting the right learning algorithm for the analyzed data. This initial assessment is time consuming since one has to decide which of the learning strategies is most suited given the context. No definite way of discovering the best learning algorithm for a new problem has been devised yet, but many proposals for selecting a good technique exist in the literature. Selecting a suitable learning algorithm for a new data set is a complex task even for an experienced data analyst. Moreover, some hidden knowledge could be present in data. Such knowledge can sometimes be surmised by the domain experts, yet not so often by the data analyst. Therefore, an initial assessment should always be performed, in order to identify the most promising knowledge extraction methodology for the given problem. The process usually involves training several models with different learning algorithms whose parameters settings vary and evaluating their performance (perhaps under several metrics) with respect to the requirements of the problem. The analyst can then choose the learning algorithm and the settings which register the best performance. The time required to build a model increases with the complexity of the model and with the size of the input data. Running and evaluating all known learning algorithms is therefore unfeasible.

A more suitable approach involves comparing the new problem with a set of problems for which the learning algorithm performance is already

Cacoveanu S., Lemnaru C. and Potolea R. (2010).
EVALUATING PREDICTION STRATEGIES IN AN ENHANCED META-LEARNING FRAMEWORK.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Artificial Intelligence and Decision Support Systems*, pages 148-156
DOI: 10.5220/0002975401480156

known (Aha,1992) (Bensusan,2000) (Giraud-Carrier,2000) (Vilalta,2004). The analyst must identify the problem which resembles the analyzed data the most. Consequently, the same learning algorithm and settings that obtained the best results on the former problem is expected to achieve similar performance on the new problem. To make use of this approach, the expert must have access to a large set of problems that have already been evaluated with various techniques. Also, the success of the selected learning algorithm on the new problem depends on the expert's strategy for selecting similar problems.

Creating a framework which brings together all the tools necessary to analyze new problems and make predictions related to the learning algorithms' performance would automate the analyst's work. This will result in a significant speed-up and an increased reliability of the learning algorithm selection process. Such a framework would prove to be valuable for an experienced data analyst and could also help new users discover a good classifier, regardless of their knowledge of the data mining domain and the problem context (for users that do not know if maximizing the accuracy or minimizing the false-positive rate is preferable in the given problem context). We have already proposed such a tool in (Cacoveanu, 2009) and developed an implementation based on classifiers provided by the Weka framework (Witten,2005). Our initial focus was on selecting a wide range of dataset features and improving the classifier prediction time. We also wanted to facilitate the addition of new datasets such that our system continuously improves its performance.

This paper presents several enhancements to a meta-learning framework we have previously designed and implemented. Its main goal is to automatically identify the most reliable learning schemes for a particular problem, based on the knowledge acquired about existing data sets, while minimizing the work done by the user but still offering flexibility.

The rest of the paper is organized as follows: In section 2 we present the meta-learning frameworks presented in literature. Section 3 presents a formal model of our tool, the characteristics we use to describe datasets and the metrics implemented in the system. Section 4 describes the prediction strategies we tested with our tool and the results. We conclude the paper by restating the improvements added to our framework and proposals for future development.

2 RELATED WORK

Aha (Aha,1992) proposes a system that constructs rules which describe how the performance of classification algorithms is determined by the characteristics of the dataset.

Rendell et al. describe in (Rendel,1987) a system called VBMS. Their system tries to predict which algorithms will perform better for a given classification problem using the problem characteristics (number of examples and number of attributes). The main disadvantage of VBMS is that it is trained as new classification tasks are presented to it, which makes it slow.

The approach applied in the Consultant expert system (Sleemann et al (Sleeman,1995)) relies heavily on a close interaction with the user. Consultant poses questions to the user and tries to determine the nature of their problem from the answers. It does not examine the user's data.

Schaffer (Schaffner,1993) proposes a brute force method for selecting the appropriate learner: execute all available learners for the problem at hand and estimate their accuracy using cross validation. His system selects the learner that achieves the highest score. This method has a high demand of computational resources.

STATLOG (Michie,1994) extracts several characteristics from datasets and uses them together with the performance of inducers (estimated as the predictive accuracy) on the datasets to create a meta-learning problem. It then employs machine learning techniques to derive rules that map dataset characteristics to inducer performance. The limitations of the system include the fact that it considers a limited number of datasets; it incorporates a limited set of data characteristics and uses accuracy as the sole performance measure.

P. B. Brazdil et al. propose in (Brazdil,2003) an instance-based learner for obtaining an extensible system which ranks learning algorithms based on a combination of accuracy and time.

3 A FRAMEWORK FOR SEARCHING FOR SUITABLE LEARNERS

This section starts by presenting a formal model for a classifier prediction framework. The framework is built over a database containing stored problems already analyzed and classifiers used in the analysis

process. The IO part of the framework consists of the problem and requirements provided by the user and the result, a selection of suitable classifiers, returned by the framework. The second part of the section describes the features we used to describe data mining problems. The last part of this section presents the metrics we use to evaluate the performance of a classification model.

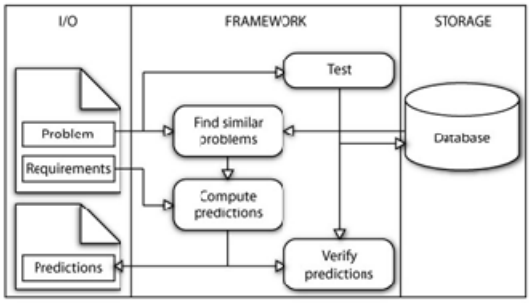

Figure 1: System diagram.

3.1 A formal Model of our Tool

In this section we present a formal model for an automated learner selection framework (Figure 1). The basic functionality of a learner selection system is to evaluate, rank and suggest an accurate learning algorithm for a new problem submitted to the system, together with the user's requirements. The suggested algorithm can then be used on the problem to induce a model which achieves the best performance while meeting the user's requirements. The user involvement is limited to providing the input problem (i.e. a data set which the user needs to classify) and specifying the requirements. The result is presented to the user as an ordering of learning algorithms, arranged in decreasing performance order. Recommending more than a single learning algorithm is important as it allows the user to decide on the specific strategy they want to follow (they might prefer a faster but least accurate algorithm to a very accurate one, or an algorithm they are accustomed to). The system should also minimize the time it takes to provide the results.

The process of obtaining the predictions is roughly divided into selecting the similar problems and obtaining predictions from similar problems. In order to be able to provide accurate predictions for new datasets the system relies on a database containing the problems and the solutions (classifier + performance) obtained on those problems. The system must have the ability to increase its knowledge by adding new problems and the corresponding solutions. To do this, the first important functionality of the system is the ability to run learning algorithms and evaluate those algorithms. This occurs mainly in the initialization phase of the system.

After a significant collection of problems has been stored in the database, the system is ready for the prediction phase. In this phase, a new problem is submitted to the system, along with its requirements. The system must now find similar problems in its database. This is done by computing the distance between the analyzed problem and the stored problems. Once the distances have been evaluated, a subset of the nearest stored problems is selected as neighbors of the analyzed problem.

The results obtained by the learning algorithms for every neighbor problem are already present in the database. The performance score of each classifier is obtained by evaluating the results obtained by that classifier from the perspective of the user requirements. The framework then predicts the performance score for the classifier on the analyzed problem as a combination of the performance scores obtained by that classifier on the neighboring problems. The final list of recommended learning algorithms is ordered by their predicted performance scores.

After a new prediction is performed, during the time the system is idle (it does not have to perform another prediction), it continues with an extension of the initialization phase. More specifically, it trains and evaluates models on each new problem added to the system and saves the new data to the database. This way, the system's knowledge increases and its prediction capabilities improve.

3.2 Problem Characterization – Meta-features

In order to estimate the similarity (i.e compute the distance) between problems (i.e. data sets), a series of meta-features are extracted from the data sets. The meta-features we employ in this framework can be divided into four categories. One of these categories is focused on the type of the attributes in the data sets. It contains the total number of attributes of a data set (Michie,1994), the number of nominal attributes (Kalousis,2002), the number of boolean attributes (Kalousis,2002) (Michie,1994) and the number of continuous (numeric) attributes (Kalousis,2002). Another category is focused on analyzing the properties of the nominal and binary attributes of the data sets. This category contains the maximum number of distinct values for nominal attributes (Kalousis,2002) (Linder,1999), the

minimum number of distinct values for nominal attributes (Kalousis,2002) (Linder,1999), the mean of distinct values for nominal attributes (Kalousis,2002) (Linder,1999), the standard deviation of distinct values for nominal attributes (Kalousis,2002) (Linder,1999) and the mean entropy of discrete variables (1) (Kalousis,2002).

$$\sum_{i=1}^{n} p(x_i) log_b(p(x_i)) \quad (1)$$

Similar to the previous category, the next category focuses on the properties of the continuous attributes the data sets have. It includes the mean skewness of continuous variables (2) (Kalousis,2002) (Michie,1994), which measures the asymmetry of the probability distribution, and the mean kurtosis of continuous variables (3) (Kalousis,2002) (Michie,1994) representing the peak of the probability distribution.

$$\gamma_1 = \frac{\mu_3}{\sigma_3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^3}{\sqrt{(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2)^3}} \quad (2)$$

$$\gamma_2 = \frac{\mu_4}{\sigma_4} = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^4}{(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2)^2} \quad (3)$$

A final category gives a characterization of the dimensionality of the data set. It contains the overall size, represented by the number of instances, and imbalance rate information (relative size) (Japkowicz,2002). The mean (5) and maximum (4) imbalance rates of the classes in the dataset are computed (in case there are only 2 classes, the mean and maximum imbalance rates are equal).

$$max_{IR} = \max(\{IR_i | i = \overline{1,c}\}), IR_i = \frac{\max(\{C_i, I - C_i\})}{\min(\{C_i, I - C_i\})} \quad (4)$$

I = number of instances, C_i=number of instances belonging to class i, c=number of classes

$$\sum_{i=1}^{c} (IR_i * \frac{C_i}{I}) \quad (5)$$

3.3 Classifier Performance Metrics

The performance score of a classifier depends on the problem requirements provided by the user. When designing the system, we have focused on minimizing its dependence of user input. We still need to provide the user with a method of guiding the search for a suitable learning algorithm. For this purpose, we employ nine metrics, divided into three categories, as proposed in (Caruana,2004). The metrics in Table 1, along with a general purpose metric are described in this section.

Table 1: Performance metrics.

Threshold	Rank	Probability
Accuracy, Recall, False Positive Rate, True Negative Rate, False Negative Rate	Area Under ROC, Precision	Geometric Mean, Generalized Geometric Mean

Most classifier performance metrics are generated from the confusion matrix produced by the induced model on a test sample. The confusion matrix is the most general indicator of the way a model identifies the right label of instances. An example of a confusion matrix for c classes is provided in Figure 2. The entry from the i[th] row j[th] column represents the number of instances from class i that were labeled by the model as belonging to class j.

$$\begin{bmatrix} M_{1,1} & \cdots & M_{1,c} \\ \vdots & \ddots & \vdots \\ M_{c,1} & \cdots & M_{c,c} \end{bmatrix}$$

Figure 2: Confusion matrix for dataset with c classes.

While most users will be concerned with the accuracy of the generated models, in some cases they might prefer to focus on improving a different performance criterion (for instance to maximize the sensitivity or specificity). This difference usually comes from the different costs associated to specific errors. As an example, the cost of an incorrect medical diagnosis is different for false positive and false negative errors.

Problem requirements are the way users set the focus of the search for a learning algorithm. Users provide problem requirements in terms of performance metrics. When a classifier's performance is measured, the resulting score is computed from the model's confusion matrix or area under the ROC curve, based on the user requirements. Both the confusion matrix of a classifier and the area under ROC are computed in the initialization phase, when evaluating the performance of a trained model (using 10-fold cross validation).

The *Accuracy* (6) of a classifier is the percentage of test instances that are correctly classified by the model (also referred to as the recognition rate). Accuracy selects learning algorithms with the highest rate of correct classification but it is a weak metric for imbalanced problems. For example, if the number of negative cases in the test sample is much larger than the positive cases, even if all positive cases are misclassified, the accuracy will still be very high.

The *Recall* (true positive rate) (7) is the proportion of positive cases that were correctly identified. This metric favors models which focus on identifying the positive cases, even if this leads them to misclassifying a number of negative cases as positive. The system also provides metrics that maximize the *false positive rate* (8), the *true negative rate* (9) and the *false negative rate* (10).

$$ACC = \frac{\sum_{i=1}^{c} M_{i,i}}{\sum_{i=1}^{c}\sum_{j=1}^{c} M_{i,j}} \quad (6)$$

$$TPR = \sum_{i=1}^{c}\left(\frac{M_{i,i}}{\sum_{j=1}^{c} M_{i,j}} * \frac{C_i}{I}\right) \quad (7)$$

$$FPR_i = \frac{\sum_{j\in\{1\ldots c\}\setminus\{i\}} M_{j,i}}{\sum_{j\in\{1\ldots c\}\setminus\{i\}}\sum_{k\in\{1\ldots c\}} M_{j,k}},\ FPR = \sum_{i=1}^{c} FPR_i * \frac{C_i}{I} \quad (8)$$

$$TNR_i = \frac{\sum_{j\in\{1\ldots c\}\setminus\{i\}}\sum_{k\in\{1\ldots c\}\setminus\{i\}} M_{j,k}}{\sum_{j\in\{1\ldots c\}\setminus\{i\}}\sum_{k\in\{1\ldots c\}} M_{j,k}},\ TNR = \sum_{i=1}^{c} TNR_i * \frac{C_i}{I} \quad (9)$$

$$FNR_i = \frac{\sum_{j\in\{1\ldots c\}\setminus\{i\}} M_{i,j}}{\sum_{j\in\{1\ldots c\}} M_{i,j}},\ FNR = \sum_{i=1}^{c} FNR_i * \frac{C_i}{I} \quad (10)$$

The *precision* is the proportion of the predicted positive cases that are correctly classified. By maximizing precision, a model rarely classifies negative cases as being positive, but may still misclassify positive cases as negative.

The *ROC* graph is a representation of the true positive rate (*sensitivity*) as a function of the false positive rate (1-*specificity*). In case of a perfect classifier, the area under the ROC curve is 1, because the false positive rate is 0 while the true positive rate is 1. This metric is usually employed when searching for a classifier that maximizes the number of correctly classified instances.

Besides the fundamental metrics the system evaluates for a data set, some combined metrics are available as well. The *geometric mean* metric (11) is used to maximize the true positive and the false positive rate at the same time.

$$\sqrt{\frac{TP}{TP+FN} * \frac{TN}{FP+TN}} \quad (11)$$

We also propose the *generalized geometric mean* – a generalization of the geometric mean metric for datasets with more than two classes (12).

$$GGM = \sqrt[c]{\prod_{i=1}^{c}\left(\frac{M_{i,i}}{\sum_{j=1}^{c} Mi,j}\right)} \quad (12)$$

For allowing users from different areas of expertise which are interested in discovering a generally good classifier, we propose a metric which combines the accuracy, the geometric mean and area under the ROC curve (13). This metric is obtained by computing the average of three other metrics, each being the best in its category, as observed in (Caruana,2004).

$$\frac{ACC + GM + AUC}{3} \quad (13)$$

4 EXPERIMENTS ON AN ENHANCED VERSION OF THE FRAMEWORK

When first proposing a framework for classifier selection in (Cacoveanu, 2009) we focused on selecting a wide range of dataset features and improving classifier prediction time. In the attempt to improve the prediction capabilities of our framework, we automated the voluntary addition of new data sets. The prediction was performed by using a KNN classifier which computed the distances between the analyzed data set and all data sets stored in the system. It then selected the three closest data sets as neighbors and estimated the predicted performance as the mean between the performances obtained by a classifier on the neighboring data sets.

As an improvement to our data set classification system, we considered the possibility of using different strategies when selecting the similar problems to the one we are analyzing and the way neighboring problems affect the final performance predictions. We are also trying to answer the question of how different strategies behave when making performance predictions for different metrics. An ideal solution would be to find a set of approaches that gives the best results for all metrics, and always use them. However, if the selection of a given tactic influences the predictions for different metrics generated by the framework, we may consider always using the best strategies for the metric selected by the user.

We have divided the classifier accuracy prediction process into three phases: distance computation, neighbor selection and prediction computation (or voting). For each of these phases we propose several strategies (detailed in the next sub-sections).

4.1 Distance Computation

The distance computation phase consists in computing the distance between the currently analyzed dataset and all the datasets stored in the system database. The distance is computed by using the data set meta-features (all having numeric values) as coordinates of the data set. By representing a data set as a point in a vector space,

the distance can be evaluated using any metric defined on a vector space (14).

$$d_i(f_1^i, f_2^i, \ldots, f_m^i), \text{ m=number of features} \quad (14)$$

The first distance computation strategy implemented is the normalized Euclidean distance (***E***). The Euclidean distance is the ordinary distance between two points in space, as given by the Pythagorean formula (15). The obtained distances are normalized in the [0,1] set.

$$Dist_E(d_x, d_y) = \sqrt{\sum_{i=1}^{m} (f_i^x - f_i^y)^2} \quad (15)$$

Another distance evaluation available in our framework is the Chebyshev distance (***C***). The Chebyshev distance is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension. In our system, the largest difference between dataset features is the distance between two datasets (16). These distances are also normalized.

$$Dist_C(d_x, d_y) = \max\left(\{abs(f_i^x - f_i^y) | i = \overline{1,m}\}\right) \quad (16)$$

The advantage of the Chebyshev distance computation strategy is that it takes less time to decide the distances between datasets. A possible problem with the Chebyshev distance is allowing one single feature to represent a dataset. One single largest feature might not offer enough description of the dataset to lead to accurate neighbor selection and final predictions. As a solution we propose a variation of the Chebyshev distance (***C3***), where the largest three differences between features are selected and their mean is computed (17). By using this strategy, the dataset information that is used in the distance computation process increases but the computation remains more efficient than the Euclidean distance.

$$\begin{gathered} F = \{abs(f_i^x - f_i^y) | i = \overline{1,m}\}, \\ max_k = \max\left(F - \{max_j | j = \overline{1,k-1}\}\right) \\ Dist_{C3}(d_x, d_y) = \frac{\sum_{l=1}^{3} max_l}{3} \end{gathered} \quad (17)$$

4.2 Neighbor Selection

Neighbor selection decides which datasets will influence the performance predictions for the analyzed dataset. While considering as neighbors the datasets closer to the analyzed dataset is justified, how many datasets should be considered is an open problem. If we choose a fixed number of neighbors we make sure the prediction computation will always take the same amount of time. We implement the Top 3 neighbor selection strategy (***T3***) which selects the 3 closest datasets as neighbors. Selecting the closest n neighbors is a sensible decision for a strategy, but there could be cases in which the closest n datasets are still quite far. Another approach would be setting a threshold distance after which a dataset is not considered a neighbor anymore. By using a fixed threshold value we risk getting into situations in which not one dataset will be considered a neighbor and we will not be able to compute the performance predictions. A solution would be to have a variable threshold depending on the average distance between every two datasets in the system at the moment a new dataset arrives, but this solution means updating the average distance every time a new dataset arrives in the system and holding a different threshold value for all the distance computation strategies in the system. We chose to implement a strategy similar to the variable threshold considered above, only this time we compute the average distance between the analyzed dataset and all the datasets in the system and select as neighbors only datasets closer than this average distance. We call this strategy Above-Mean neighbor selection (***AM***). This strategy selects a variable number of neighbors every time and the performance predictions computation time increases considerably.

4.3 Prediction Computation (Voting)

In this last phase of the prediction process the classifier performance predictions are generated. Voting strategies define the way neighboring datasets influence the final predictions. Each neighbor casts a performance score to each classifier in the system. The performance score for a classifier depends on the performance of its model on the dataset (the confusion matrix, the area under ROC) evaluated from the point of view of the metric selected by the user. These scores are combined to obtain the final score for each classifier. This final score is the actual predicted performance for that classifier. Classifiers are then ordered based on the predicted performances.

We have implemented two voting strategies in the system, the first one of them being equal, or democratic, voting (***D***). Each neighbor selected in the previous phase predicts a performance for the current classifier. We sum all the performances and divide them by the number of neighbors (18). The result is the predicted performance of the current classifier. Each neighbor has the same influence in deciding the final performance of a classifier.

$$P_c = \frac{\sum_{i=1}^{n} P_c^i}{n},$$

n = number of neighbors

P_c^i= performance obtained on dataset i, with classifier c,

P_c = predicted performance for classifier c (18)

The second voting strategy is weighted voting (***W***). For this, the distances between the analyzed dataset and its neighbors act as the weight of the neighbor vote – a closer dataset will have a higher weight and more influence on the final prediction (19).

$$w_i = \frac{1 - dist(d, d_i)}{\sum_{j=1}^{n}(1 - dist(d, d_i))} \quad (19)$$

$P_c = \sum_{i=1}^{n} P_c^i * w_i$, d= analyzed dataset, d_i= neighbor i

4.4 Experimental Results

This section presents the results of the evaluations performed with our framework to find the combination of strategies that works best in predicting performance scores for the learning algorithms in the system.

We initialized our system with 26 benchmark datasets that range from very small to medium sizes (up to 4000 instances) (UCI, 2010). Also, the following classifiers are available: BayesNet, J48, MultilayerPerceptron, AdaBoost, NaiveBayes, SMO, PART, libSVM.

We have performed evaluations with all the possible combinations of the implemented strategies for distance computation, neighbor selection and performance score prediction (Table 2).

Table 2: Strategy combinations.

Distance computation strategy	Neighbor selection strategy	Voting strategy	Notation
E	T3	D	E-T3-D
E	T3	W	E-T3-W
E	AM	D	E-AM-D
E	AM	W	E-AM-W
C	T3	D	C-T3-D
C	T3	W	C-T3-W
C	AM	D	C-AM-D
C	AM	W	C-AM-W
C3	T3	D	C3-T3-D
C3	T3	W	C3-T3-W
C3	AM	D	C3-AM-D
C3	AM	W	C3-AM-W
E = Normalized Euclidean distance C = Normalized Chebyshev distance C3 = Normalized Top 3 Chebyshev distance			
T3 = Top 3 neighbor selection AM = Above Mean neighbor selection			
D = Equal (Democratic) voting W = Weighted voting			

For a test, we selected a performance metric and executed the following steps:

1. Selected a strategy combination
   a. Selected a dataset and used it as the analyzed dataset
      i. Used the remaining 25 datasets as datasets stored in the system
      ii. Used the selected strategy combination to predict performances
      iii. Compare the predicted performances with the actual performances obtained in the initialization stage on the selected dataset
   b. Select next dataset
2. compute the deviation mean and the absolute deviation mean on all datasets and classifiers for this strategy
3. select next strategy combination

We have applied the above strategy for the following metrics: accuracy, geometric mean, generalized geometric mean, area under ROC, general purpose metric.

In total, we ran 312 prediction tests for each selected metric.

We have computed the deviation between the predicted and true performance as the difference between the performance prediction and the actual performance (20). If the system predicted a classifier will obtain a higher performance than it actually obtains, this value will be negative.

$$D = P_a - P_p$$
$$D_{abs} = |P_a - P_p| \quad (20)$$

P_p=performance prediction, P_a=actual performance

The absolute deviation between a performance prediction and the actual performance is the absolute value of the difference between the two (20).

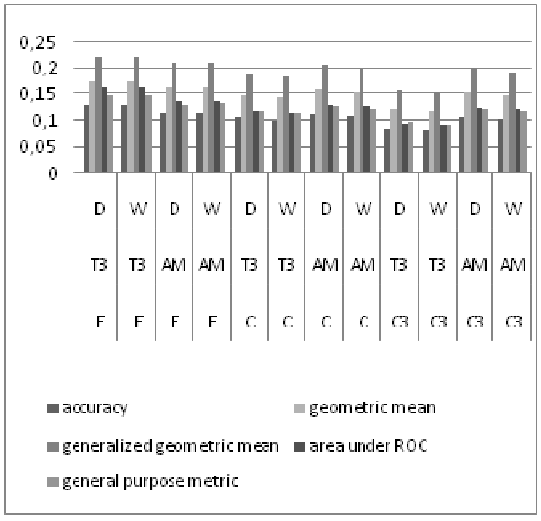

Figure 3: Absolute deviation mean.

Our interest is to select the strategies that minimize the deviation means for all the metrics. By studying the absolute deviation mean results (Figure 3), we observe that voting strategies do not influence the final predictions very much. Weighted voting (W) obtains better results than democratic voting (D), but the difference is so small that it does not justify the additional resources needed to compute the weight of each neighbor. Moreover, we observe that the distance computation and neighbor selection strategies that obtain the smallest absolute deviations are, in order: (1) Top 3 Chebyshev distance with Top 3 neighbor selection, (2) Chebyshev distance with Top 3 neighbor selection and (3) Chebyshev distance with Above Mean neighbor selection.

By analyzing the deviation mean results table (Figure 4) we can deduce more details on how each of the three selected strategy combinations work. Our first choice, Top 3 Chebyshev distance with Top 3 neighbor selection, has negative deviation means for all the metrics. From this we deduce that the strategy combination is overly optimistic – most of the time it will predict performances that are not met when we derive the model and evaluate it. We would prefer that our system slightly underestimates the performance of a model on a new dataset. We can observe that the second strategy combination, Chebyshev distance with Top 3 neighbor selection, makes optimistic

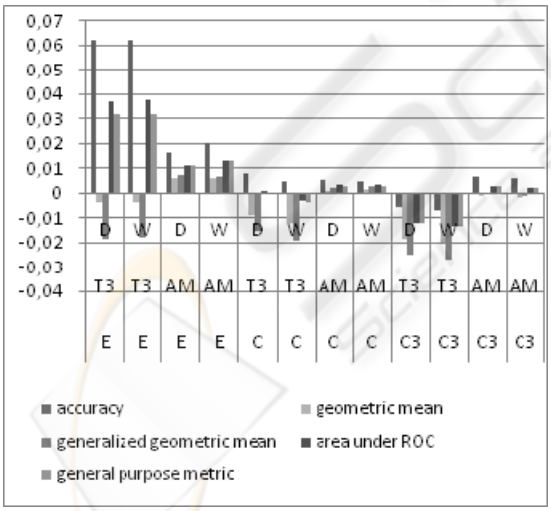

Figure 4: Deviation mean.

predictions for all metrics except accuracy. This strategy combination is the best choice when predicting classifier accuracy, but is not appropriate for the other metrics in the system. The last strategy combination, Chebyshev distance with Above Mean neighbor selection, obtains positive deviation means for all metrics. This is the preferred behavior for our system and we can conclude that this is the best combination of strategies.

We can observe from both Figure 4 and Figure 3 that the deviation mean of the general purpose metric is close to the average deviation means of the other metrics. Therefore, we can confirm the conclusion in (Caruana,2004) that a general-purpose metric has the best correlation with the other metrics.

5 CONCLUSIONS AND FUTURE WORK

This paper describes the architecture of an automated learner selection framework. We focus on the enhancements considered and tested for the system described in (Cacoveanu, 2009). These enhancements consist in increasing the dataset pool, adding new performance metrics and meta-features and improving the prediction accuracy of the system. The increase in metrics widens the evaluation criteria and allows a more problem-specific assessment of classifiers. Two of the newly added metrics, the generalised geometric mean and the general purpose metric, both of them representing original proposals. Moreover, the general-purpose metric proposed has suggested a new approach in dealing with data sets inputs with no associated metrics. Another enhancement was the addition of new benchmark data sets. The increase in the data available to the system improves the outcome of the neighbor estimation step. We also implemented the context for adding complex prediction strategies. We implemented and evaluated 12 strategy combinations for computing the final performance predictions for classifiers. The analysis of the results suggest as a best strategy the Chebyshev distance with Above Mean neighbor selection and Democratic voting. This strategy will predict performances close to the actual performances, without surpassing them. The tests also reveal that the voting strategies do not significantly influence the final results. Moreover, in the case of the accuracy metric we can improve the performance of our system by using Chebyshev distance with Top 3 neighbor selection and Democratic voting.

Since the Chebyshev distance computation strategy obtains the best results in our system, our present focus is on discovering the most relevant data set features, by performing feature selection on

the meta-data set or deriving new and more relevant features (Niculescu-Mizil, 2009). We attempt to improve the Above Mean neighbor selection strategy, by computing and constantly updating a mean distance between every two datasets in our database. Limiting the neighbor selection strategy as the number of problems in the system increases is another present concern. We also want to improve the system by generating a "best-possible" model. For this we intend to use different classifiers, each classifier optimized to increase the true positive rate on its class, thus maximizing the prediction power of the model.

ACKNOWLEDGEMENTS

Research described in this paper was supported by the IBM Faculty Award received in 2009 by Rodica Potolea from the Computer Science Department of the Faculty of Computer Science, Technical University of Cluj-Napoca, Romania.

REFERENCES

Aha D. W., 1992, Generalizing from Case Studies: A Case Study, *Proceedings of the Ninth International Conference on Machine Learning*, pp. 1 -10

Bensusan H., Giraud-Carrier C., Kennedy C. J., 2000, A Higher-Order Approach to Meta-Learning, *proceedings of the ECML-2000 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pp. 33-42

Brazdil P. B., Soares C., Da Costa J. P., 2003, Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results, *Machine Learning 50*, pp. 251-277

Cacoveanu S., Vidrighin C., Potolea R., 2009, Evolutional Meta-Learning Framework for Automatic Classifier Selection, *proceedings of the 5th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca*, pp. 27-30

Caruana R., Niculescu-Mizil A., 2004, Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 69-78

Giraud-Carrier C., Bensusan H., 2000, Discovering Task Neighbourhoods Through Landmark Learning Performances, *Proceedings of the Fourth European Conference of Principles and Practice of Knowledge Discovery in Databases*, pp. 325-330

Japkowicz N., Stephen S., 2002, The Class Imbalance Problem: A Systematic Study, *Intelligent Data Analysis, Volume 6, Number 5,* pp. 429-450

Kalousis A., 2002, Algorithm Selection via Meta_Learning, *PhD Thesis, Faculte des sciences de l'Universite de Geneve*

Linder C., Studer R., 1999, Support for Algorithm Selection with a CBR Approach, *Proceedings of the 16th International Conference on Machine Learning*, pp. 418-423

Michie D., Spiegelhalter D. J., Taylor C. C., 1994, Machine Learning. Neural and Statistical Classification, *Ellis Horwood Series in Artificial Intelligence*

Niculescu-Mizil A., et al, 2009, Winning the KDD Cup Orange Challenge with Ensemble Selection, *JMLR Workshop and Conference Proceedings 7*

Rendel L., Seshu R., Tcheng D., 1987, Layered concept learning and dynamically variable bias management, *10th Internatinal Joint Conf. on AI*, pp. 308-314

Schaffner C., 1993, Selecting a classification method by cross validation, *Machine Learning 13*, pp. 135-143

Sleeman D., Rissakis M., Craw S., Graner N., Sharma S., 1995, Consultant-2: Pre and post-processing of machine learning applications, *International Journal of Human Computer Studies*, pp. 43-63

UCI Machine Learning Data Repository, http://archive.ics.uci.edu/ml/, last accessed Jan. 2010

Vilalta R., Giraud-Carrier C., Brazdil P., Soares C., 2004, Using Meta-Learning to Support Data Mining, *Iternational Journal of Computer Science & Applications*, pp. 31-45

Witten I. H., Frank E., 2005, Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition, *Morgan Kaufmann Publishers, Elsevier Inc.*