A Novel Multiple Classifier Generation and Combination Framework Based on Fuzzy Clustering and Individualized Ensemble Construction

Zhen Gao
Department of Computer Science
University of Texas at San Antonio
San Antonio, United States
zhen.gao@utsa.edu

Maryam Zand
Department of Computer Science
University of Texas at San Antonio
San Antonio, United States
maryam.zand@utsa.edu

Jianhua Ruan*
Department of Computer Science
University of Texas at San Antonio
San Antonio, United States
Jianhua.Ruan@utsa.edu

Abstract—**Multiple classifier system (MCS) has become a successful alternative for improving classification performance. However, studies have shown inconsistent results for different MCSs, and it is often difficult to predict which MCS algorithm works the best on a particular problem. We believe that the two crucial steps of MCS - base classifier generation and multiple classifier combination, need to be designed coordinately to produce robust results. In this work, we show that for different testing instances, better classifiers may be trained from different subdomains of training instances including, for example, neighboring instances of the testing instance, or even instances far away from the testing instance. To utilize this intuition, we propose Individualized Classifier Ensemble (ICE). ICE groups training data into overlapping clusters, builds a classifier for each cluster, and then associates each training instance to the top-performing models while taking into account model types and frequency. In testing, ICE finds the k most similar training instances for a testing instance, then predicts class label of the testing instance by averaging the prediction from models associated with these training instances. Evaluation results on 49 benchmarks show that ICE has a stable improvement on a significant proportion of datasets over existing MCS methods. ICE provides a novel choice of utilizing internal patterns among instances to improve classification, and can be easily combined with various classification models and applied to many application domains.**

Index Terms—**Classification, Multiple classifier system, Ensemble Learning**

I. INTRODUCTION

Multiple classifier system (MCS), including ensemble classifiers and mixture of experts, has established itself as an effective and practical solution to address challenges in supervised learning, such as functional complexity, insufficient training data, high dimensionality of feature space, and noise in training data, among others. Many excellent comprehensive reviews on MCS algorithms are available [1]–[3].

Learning a MCS usually includes two critical steps: base classifier generation, and multiple classifier combination, although sometimes the two steps are intrinsically integrated. Different MCS methods can be distinguished by how these two steps are performed. According to model generation strategies, existing MCS methods usually fall into one of the following two categories: random methods and deliberate methods. The former generates models by injecting random perturbations into the training data or training process [4], [5]. In contrast, the latter attempts to generate multiple classifiers in a more systematic, principled way, e.g., by iteratively re-weighting the training instances with emphasis on previously misclassified instances, a technique known as boosting [6], or by first clustering the training instances and then learning submodels from each cluster [7], [8]. According to model combination strategies, MCS methods can also be grouped into two categories: voting-based and learning-based. Most popular ensemble methods (e.g., bagging and boosting) take a (weighted) voting from all models in the pool. Other methods attempt to learn a high-level model in order to determine which model(s) should be selected for the prediction task, or to learn a more complex function to combine the outputs of all models in the pool. Learning-based model combination algorithms include stacking, dynamic model selection, among many others [9], [10].

Overall, ensemble approaches combining randomized model generation and voting (e.g. bagging and random forest) have been more successful / popular, probably due to their simplicity and less over-fitting. On the other hand, it has been shown that careful integration of deliberate models and learning-based model combination can be very effective on specific problem domains [11]. In particular, empirical studies suggest that many classification problems consist of subdomains, which can potentially benefit from constructing and selecting submodels [7], [8], [12]. The challenge, however, lies in whether these subdomains can be corrected identified at training, and whether the submodels can be correctly selected for individual cases at prediction time.

Here, we design a general MCS framework, Individualized Classifier Ensemble (ICE), with two key ideas. First, it constructs a large pool of submodels that have low bias when applied to appropriate instances. This is achieved by applying a strong learner (in contrast to the high-bias, low-variance models commonly used in a few ensemble methods) to individual overlapping clusters of instances that represent
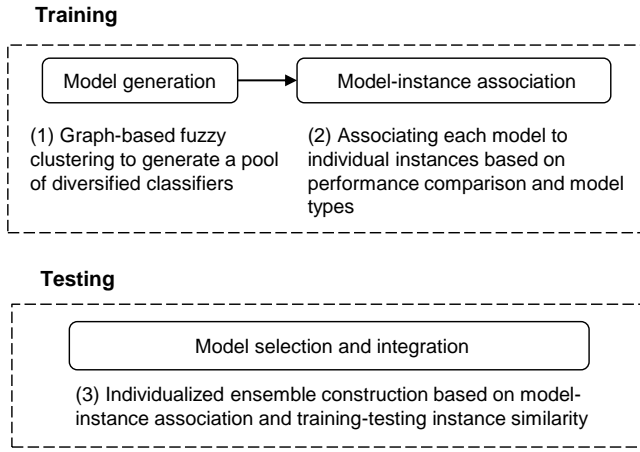
Fig. 1. **Overview of the ICE framework.**

Algorithm 1 Training

1: **procedure** TRAIN ($\boldsymbol{X}$, $\boldsymbol{Y}$, L, w, s)
2:     $\boldsymbol{C} \leftarrow$ CLUSTERING ($\boldsymbol{X}, L$)
3:     $\boldsymbol{O} = \{o_i\}_{i=1}^{L} \leftarrow \emptyset$
4:     **for** c_j in $\boldsymbol{C}$ **do**
5:         $o_j \in \boldsymbol{O} \leftarrow$ Train model on c_j using base-classifier
6:     **end for**
7:     $\boldsymbol{D} \leftarrow$ ASSOCIATE ($\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{C}, \boldsymbol{O}, w, s$)
8:     **return** $\boldsymbol{D}, \boldsymbol{O}$
9: **end procedure**

Algorithm 2 Graph-based fuzzy clustering

1: **procedure** RWFCLUSTERING ($\boldsymbol{X}$, L)
2:     $\boldsymbol{S}_{Q \times Q} \leftarrow$ Euclidean dist. based inst. similarities on $\boldsymbol{X}$
3:     $\boldsymbol{G}_{Q \times Q} \leftarrow$ Each node keeps top $\lceil log_{10}Q \rceil$ Nbr. on $\boldsymbol{S}$
4:     $\boldsymbol{W}_{Q \times Q} \leftarrow$ Random walk with restart on $\boldsymbol{G}$
5:     $\boldsymbol{T} = \langle t_j \rangle_{(L-1) \times 1} \leftarrow \langle 0 \rangle_{(L-1) \times 1}$
6:     $t_1 \leftarrow$ find the most connected node in $\boldsymbol{W}$
7:     **for** j in $2...(L-1)$ **do**
8:         $t_j \leftarrow$ find the common farthest node of $\boldsymbol{T}$
9:     **end for**
10:     $\boldsymbol{R} = \langle r_{ij} \rangle_{Q \times Q} \leftarrow$ keep top z edges in $\boldsymbol{W}$
11:     $\boldsymbol{C} = \{c_i\}_{i=1}^{L} \leftarrow \emptyset$
12:     **for** j in $1...(L-1)$ **do**
13:         $c_j \leftarrow$ find indices of $r_{t_j \bullet} == 1$
14:     **end for**
15:     $c_L \leftarrow 1...Q$
16:     **return** $\boldsymbol{C}$
17: **end procedure**

possible subproblems. Second, a simple yet effective, learning-free method is used to obtain different combinations of submodels for different testing instances. The learning-free nature of the method reduces the chance of selecting wrong models, therefore ensures that the combination of the selected submodels is better than, or at least no worse than, an average of all submodels.

Experimental results on 49 datasets from different domains show that ICE consistently outperforms the competing methods. Furthermore, detailed component analysis shows that both steps of our algorithm have positive contributions as expected. In addition, analysis of the submodels can shed light on the internal structure of the problem, which can potentially be used to further increase prediction performance, or to improve mechanistic understanding of the problem. The framework can be easily combined with existing classifiers and applied to many domains.

II. METHODS

Fig. 1 shows a brief overview of ICE, which starts with generating a pool of diverse and subdomain-representative classifiers from subsets of training instances (Algorithm 1), obtained by a graph-based clustering method that can detect overlapping clusters (Algorithm 2). Then, these classifiers are associated with individual training instances based on their relative prediction performance on the instance, taking into account model types and frequency (Algorithm 3). In testing/prediction stage, the nearest neighbors of a test instance are identified from the training dataset and the classifiers associated with these neighboring instances are selected to form an ensemble for prediction (Algorithm 4). While the general ICE framework is flexible and the individual components can be re-designed with domain-specific information, several design principles are crucial and are discussed below.

Source code and data are available at https://github.com/ ds-utilities/ICE.

A. Training

1) Basic notations: We define a dataset of Q training instances as $\boldsymbol{A} = \{(x_i, y_i)\}_{i=1}^{Q}$, where $x_i \in \boldsymbol{X}$ is an R dimensional feature vector and $y_i \in \boldsymbol{Y}$ is the binary label of instance i. The clustering result on $\boldsymbol{X}$ is denoted as $\boldsymbol{C} = \{c_i\}_{i=1}^{L}$; c_i is the ith cluster; L is the total number of clusters. Here we designate the last cluster c_L of $\boldsymbol{C}$ to be the whole set of instances. Without loss of generality, we assume the class labels are binary.

2) Graph-based Fuzzy Clustering: As clustering can be subjective and unstable, we recommend generating a large number of relatively independent but overlapping clusters. In addition, each cluster needs to have a sufficient number of instances to learn a strong submodel for that subdomain. In our design, we use a graph-based clustering algorithm that chooses a set of furthest points to initiate a random walk process and use probability cutoffs to control cluster size (Algorithm 2).

The algorithm works as follows. We first calculate an instance-instance distance matrix on $\boldsymbol{X}$ by Euclidean distance and store it in $\boldsymbol{S}$. Then, we construct a *KNN* graph $\boldsymbol{G}$ by keeping the top $\lceil log_{10}Q \rceil$ neighbors for each node in $\boldsymbol{S}$. Afterwards, a random walk with a restart probability p (default to 0.3 in this work) is performed on the *KNN* graph $\boldsymbol{G}$ to obtain an affinity matrix, $\boldsymbol{W}$ [13]. Next, a set of points, $\boldsymbol{T} = \langle t_j \rangle_{(L-1) \times 1}$, is identified as cluster centers: from $\boldsymbol{W}$, the node with the largest total incoming probability, t_1, is chosen as the center point of the first cluster; cluster centers for the other clusters are selected by finding the furthest node from

the current center points. Finally, a probability cutoff is applied on W to identify direct neighbors of each cluster center as members of the cluster, such that the average cluster size is z ($z = Q/3$ as default). We designate the last cluster c_L of C to be the whole set of instances. A classifier is built using instances from each cluster.

Algorithm 3 Associate instances with models by calculating the decision table

---

 1: **procedure** ASSOCIATE (X, Y, C, O, w, s)
 2: $\quad P = \langle p_{ij}\rangle_{Q\times L} \leftarrow \langle 0\rangle_{Q\times L}$
 3: $\quad E = \langle e_{ij}\rangle_{Q\times L} \leftarrow \langle 0\rangle_{Q\times L}$
 4: $\quad$ **for** c_j in C **do**
 5: $\qquad \langle p_{\bullet j}\rangle_{Q\times 1} \leftarrow$ Pred. Y with Cross Val. (o_j, c_j, X)
 6: $\qquad \langle e_{\bullet j}\rangle_{Q\times 1} \leftarrow |\langle p_{\bullet j}\rangle_{Q\times 1} - Y|$
 7: $\quad$ **end for**
 8: $\quad$ **for** $\langle e_{i\bullet}\rangle_{1\times L}$ in E **do**
 9: $\qquad e_{iL} \leftarrow (e_{iL} - w)$
10: $\qquad e_{ij} \leftarrow (e_{ij} - s)$ if $x_i \in c_j$
11: $\quad$ **end for**
12: $\quad D = \langle d_{ij}\rangle_{Q\times L} \leftarrow \langle 0\rangle_{Q\times L}$
13: $\quad$ **for** d_{ij} in D **do**
14: $\qquad d_{ij} \leftarrow 1$ if $e_{ij} \leq e_{iL}$, 0 otherwise
15: $\quad$ **end for**
16: $\quad$ **return** D
17: **end procedure**

---

3) Associating models to instances: Incorrect model selection can significantly degrade the performance of the algorithm compared to simply averaging all submodels. When the number of training instances is relatively small, supervised learning based model selection tends to overfit. Therefore, we propose a robust learning-free method (Algorithm 3), which performs model-instance association at training time and KNN-based model selection at prediction time. Importantly, the model-instance association step takes a Bayesian approach by using different cutoffs for different types of submodels, which reflects their frequency in the pool and the probability for them to outperform other types of submodels.

Formally, given the clustering result on instances, $C = \{c_i\}_{i=1}^{L}$, where c_L is the whole set of instances, the corresponding set of models built on the clusters by a base learner (e.g., SVM) is denoted as $O = \{o_i\}_{i=1}^{L}$. Here we call a model $o_i, i \in [1, L-1]$ as a '*partial*' model, since each model is built on a subset of the training instances, and, we call model o_L as the '*whole*' model, which is built on the whole set of instances. The class probabilities predicted by all models are stored in $P = \langle p_{ij}\rangle_{Q\times L}$; p_{ij} is the predicted class probability for instance i by model j; p_{iL} is the prediction probability for instance i by model built on the whole set of training instances. Note that if instance i is NOT a member of cluster c_j (in which case, we call model o_j to be a '*remote*' model of instance i), the model is directly used to predict p_{ij} for instance i; on the other hand, if instance i is a member of cluster c_j (in which case we call model o_j a '*local*' model of instance i), the value p_{ij} is obtained by 10-fold cross-validation using instances in this cluster. This process ensures that the performance evaluation used for model-instance association is not inflated, as an instance is never evaluated by a model that used the instance in training. Importantly, by not having any designated validation dataset, we are able to keep as many instances as possible for training, an important feature for small training data.

The prediction error table, $E = \langle e_{ij}\rangle_{Q\times L}$ is derived from P; $e_{ij} = |p_{ij} - y_i|$ is the prediction error for instance i by model o_j. Each row of E, $e_{i\bullet}$, represents the prediction error of different models on instance i. Given the empirical results that *local* models usually work slightly better than *whole* model and *remote* models, as well as the fact that there are more *remote* models than *local* models in the pool, we introduce two parameters to easily balance the proportion of *local*, *whole* and *remote* models in the ensemble: w as the advantage score of the *whole* model, and s the advantage score of each *local* model. Usually $s > w > 0$ to promote the inclusion of *local* models and demote *remote* models, unless the error in a remote model is significantly smaller than in the *whole* model. Each row of E is adjusted such that $e_{iL} \leftarrow (e_{iL} - w)$, and, $e_{ij} \leftarrow (e_{ij} - s)$ if $x_i \in c_j$. Then, the decision table, $D = \langle d_{ij}\rangle_{Q\times L}$, $d_{ij} \in \{1, 0\}$, where $d_{ij} = 1$ indicates association between model o_j and instance i, is derived from the error table E, by

$$d_{ij} = \begin{cases} 1, & \text{if } e_{ij} \leq e_{iL} \\ 0, & \text{otherwise} \end{cases}$$

B. Testing / prediction

Algorithm 4 Testing

 1: **procedure** PREDICT (x_t, X, D, O, N, α, β)
 2: $\quad K^{nb} = \langle k_i\rangle_{N\times 1} \leftarrow$ select N nearest Nbr. of x_t
 3: $\quad O^{nb} \leftarrow \emptyset$
 4: $\quad$ **for** k_i in K^{nb} **do**
 5: $\qquad J \leftarrow \langle d_{k_i\bullet}\rangle_{1\times L} = 1$
 6: $\qquad O^{nb} \leftarrow O^{nb} \cup O_J$
 7: $\quad$ **end for**
 8: $\quad p^{whole} \leftarrow$ Predict by base-classifier (o_L, x_t)
 9: $\quad M \leftarrow length(O^{nb})$
10: $\quad P^{partial} = \left\langle p_i^{partial}\right\rangle_{M\times 1} \leftarrow \emptyset$
11: $\quad$ **for** o_i in O^{nb} **do**
12: $\qquad p_i^{partial} \leftarrow$ Predict by base-classifier (o_i, x_t)
13: $\quad$ **end for**
14: $\quad p^t \leftarrow$ Equation 1($P^{partial}, p^{whole}, M, N, \alpha, \beta$)
15: $\quad$ **return** p^t
16: **end procedure**

For a test instance x_t, ICE first finds its N nearest neighbors from the training dataset, then predicts its class label y_t by averaging the class probabilities predicted by the models associated with the neighbor training instances (Algorithm 4). Formally, the *PREDICT*() algorithm first selects N nearest neighbors of x_t from X, and stores the indices of the neighbor instances in $K^{nb} = \langle k_i\rangle_{N\times 1}$. Then, for each neighbor instance k_i, the algorithm looks up in the corresponding decision table $d_{k_i\bullet}$ to find the models associated with the neighbor instance, and stores the associated '*partial*' models of x_t in O^{nb}. The number of '*partial*' models in O^{nb} is denoted as M. Note that although $o_i^{nb} \in O$, O^{nb} is not a subset of O,

since $\boldsymbol{O}^{nb}$ may contain duplicated models. Then we denote $\boldsymbol{P}^{partial} = \left\langle p_i^{partial} \right\rangle_{M \times 1}$ as the '*partial*' model predictions, and each $p_i^{partial}$ is predicted by o_i on x_t. The predicted class probability by the whole model is denoted as p^{whole}. Then the predicted class probability of x_t is calculated by:

$$p^t = \frac{\sum\limits_{i=1}^{M} p_i^{partial} + (\alpha M + \beta N) \cdot p^{whole}}{(\alpha + 1)M + \beta N}, \qquad (1)$$

where α is the parameter to balance the weight of '*partial*' models and the '*whole*' model; β is the parameter to adjust the weight of '*whole*' models based on the number of top neighbors to ensure at least one p^{whole} will be used in case there is no '*partial*' model.

In our experiments, α and β are both set to 1 and N is set to 5, except in cases that we vary them to analyze the contribution of different components and the robustness of our algorithm's performance.

C. Relationship with Existing MCS Methods

ICE differs from most existing ensemble methods significantly in both model generation and model combination. Popular ensemble methods such as Bagging and Random Forest generate submodels using random subsets of data, and combine them using voting. In order for these methods to work effectively, a large number of submodels is needed to reduce overall bias. In contrast, ICE generates submodels to deliberately increase model diversity by clustering training instances. A carefully designed model-instance association algorithm helps identify the best ensemble for individual instances at prediction time. On the other hand, boosting generates submodels that focus on different groups of training instances, where grouping of instances is done implicitly by iterative re-weighting and therefore lack a global view of instance space. In addition, since there is no model selection at prediction time, boosting tend to overfit in the presence of noisy training instances.

Mixtures of experts is a class of neural network models attempting to simultaneously learn multiple submodels as well as a gating function that assigns each instance to one or more submodels [14], [15] . With similar idea, several methods use clustering as a preprocessing step for classification [7], [8]. These algorithms force each instance to be in a disjoint cluster, which reduces the number of instances at training time. In addition, prediction is done only by cluster-specific models so the cost of incorrect model selection is high. Empirical results presented in the original papers show mixed performance when compared to other MCS algorithms [7], [8], [14].

Finally, a series of methods have been developed recently under the common name 'dynamic model selection' [10], [16], [17], [17]–[21]. These approaches take an ensemble of base classifiers (e.g, from bagging), then attempt to learn a high-level classification model using, for example, instance-instance similarities and model-model correlations, as input
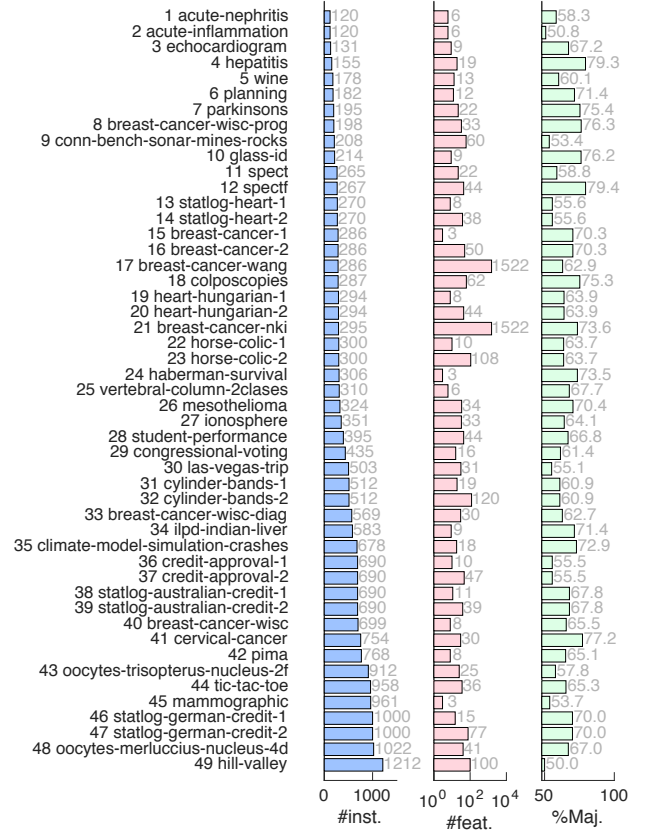

Fig. 2. **Characteristics of datasets used in evaluation.** The three columns show number of instances, number of features and percentage of majority class, respectively.

features. While conceptually appealing, these methods tend to overfit and have poor performance when training data is limited. In our opinion, the marriage between random model generation and learning-based model combination is a poor choice, since the relatively small number of random models (compared to the possible number of instance combinations) does not guarantee that there is necessarily any *predictably* better submodel than a simple average of all submodels.

III. RESULTS AND DISCUSSION

A. Data and Experimental Setup

Characteristics of the 49 benchmark datasets are shown in Figure 2. The datasets are collected from UCI machine learning repository and Kaggle Dataset for binary classification, with number of instance between 100 and 3,000, number of features between 3 and 1500, and percentage of majority class ranging from 50% to 77%. A total of 42 datasets from UCI and 23 datasets from Kaggle meet the criteria (18 of which appeared in both repositories). In addition, we add two cancer-related datasets - breast-cancer-nki [22] and breast-cancer-wang [23]. The data preprocessing mainly follows [24], which includes a Z-Score transformation based normalization. For nine datasets with nominal features we use two different methods to handle nominal features: (i) removing nominal
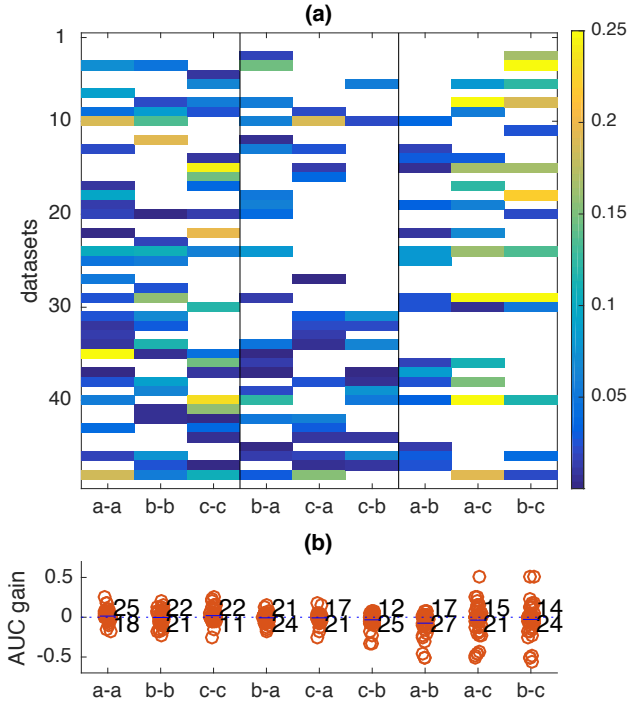
Fig. 3. **AUC gain of** *partial* **models over** *whole* **model.** (a) Color scale represents AUC gain; AUC gain ≤ 0 is removed to emphasize on potential benefits of using subsets of training instances. (b) each node represents the AUC gain on a dataset using a cluster of training data to predict a testing cluster. The two numbers on the right of each column are the number of datasets with AUC gain > 0 and AUC gain < 0.

features (denoted with suffix '-1' in Figure 2), (ii) using One-Hot encoding (denoted with suffix '-2' in Figure 2). Since all features in dataset 'tic-tac-toe' are nominal, this dataset only has the One-Hot encoding version. Data and source code are available at https://github.com/ds-utilities/ICE.

Performance of each classification method is evaluated by 10-fold cross validation and measured by AUC. To facilitate a simple and fair evaluation, we use common parameter values for ICE on all datasets. The number of overlapping clusters, L, is set to 100, which while not ideal for all data sets, makes evaluation easier. The advantage scores for '*whole*' model and '*local*' model are set to $w = 0.4$ and $s = 0.5$ respectively; this reflects the empirical observation that *local* models usually have better performance than the other two types of models, and there are many *remote* models so a higher cutoff score is needed for a *remote* model to be associated with an instance. In prediction stage, the number of top neighbors parameter N is set to 5; the parameter α and β are both set to 1 for an overall balanced '*partial*' and '*whole*' models in the final weighting of prediction. The base model in the evaluation is linear-SVM with the regularization parameter C=1 for ICE and comparison methods Bagging and AdaBoost. Bagging and AdaBoost use 100 bags and 100 iterations respectively. It is worth noting that these parameters are chosen intuitively without extensive tuning. Parameter analysis results show that the performance of ICE is robust with regarding to a wide range.

B. Empirical Evidence Supporting Cluster-Based Ensemble Classification

To verify our assumption that, for each testing instance, some subset of training instances may provide a better classification model than the whole set of training instances, we perform a simple experiment as follows: first, each dataset is clustered into three disjoint clusters using *k*-means. We denote the clusters as cluster-*a*, *b* and *c* respectively, with their cluster size decreasing. Then using instances in each cluster for cross-testing: we compared the prediction AUC for each cluster using instances from cluster *a*, *b*, *c* or the whole dataset, respectively, as training data. We adopted notation *a*-*b* to denote the situation where we use the cluster *a* trained model to make predictions on cluster *b* instances.

To have a fair evaluation, when using a larger cluster to predict a smaller cluster, we randomly select the same number of instances from the larger cluster as the size of the smaller cluster to be the training data; when use a smaller cluster to predict a larger cluster, we use all instances in the smaller cluster and randomly select some instances from the larger cluster (making sure that they are not in the fold of testing) to be the training instances, such that the total number of training instances is the same as the number of instances in the larger cluster.

From Figure 3, with only three disjoint clusters, in more than 80% of the datasets, at least one of the *local* models can outperform the *whole* model (Figure 3a and b, columns *a*-*a*, *b*-*b* and *c*-*c*). Interestingly, while in general the *remote* models do not perform well, some of them have the largest performance gain compared to the *whole* model (column *a*-*c* and *b*-*c*). Collectively, this experiment shows the potential benefit of using a cluster of instances to improve prediction accuracy. On the other hand, the results also signifies the importance to predict, for each test instance, whether *partial* models (and which) should be used.

C. ICE Outperforms Existing MCS Algorithms

Figure 4 shows that ICE outperforms the corresponding Bagging classifier on most datasets, and, suffers from only minor performance loss on a few datasets. Notably, ICE uses less than 100 base models - on average 45 models per prediction. ICE may still have room for improvement on failed datasets by parameter tuning and improved clustering methods. Understandably, ICE tends to have less performance gain on datasets with fewer instances, such as on datasets 1 to 6, since ICE needs more enriched instance information for a meaningful clustering. From another perspective, ICE will have advantage on datasets with more instances and with more complex instance structure.

Table 1 shows the complete AUCs of three versions of ICE (with SVM, Bagging and AdaBoost as the base model) on 49 benchmark datasets compared to multiple MCS methods, including Bagging, Adaboost, and seven dynamic model selection approaches. META-DES [16] has two versions in this evaluation, using Perceptron (the base classifier choice of the original META-DES paper) and Bagging (comparable with

TABLE I
AUC OF ICE AND COMPETING METHODS ON 49 DATASETS

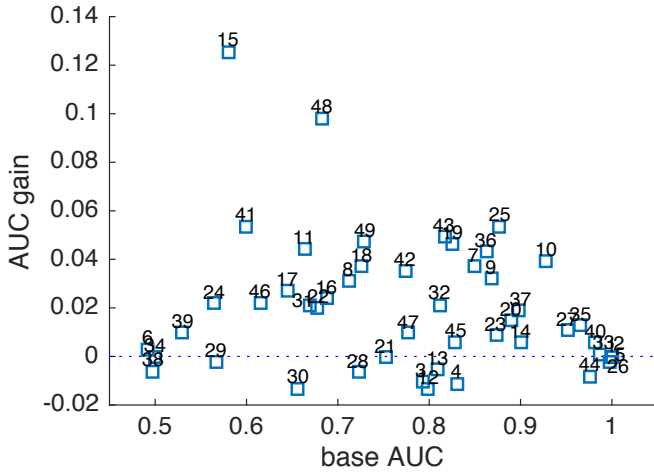|	1 META-DES Perceptron	2 META-DES Bagging	3 KNORA-U	4 KNORA-E	5 DES-PRC	6 OLA	7 MCB	8 A Priori	9 Bagging	10 AdaBoost	11 ICE	12 ICE-Bagging	13 ICE-AdaBoost
1 acute-nephritis	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2 acute-inflammation	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
3 echocardiogram	0.756	0.757	0.768	0.780	0.768	0.768	0.699	0.744	0.792	0.820	0.768	0.781	0.842
4 hepatitis	0.570	0.576	0.568	0.701	0.595	0.724	0.603	0.673	0.830	0.812	0.802	0.819	0.809
5 wine	0.930	0.941	0.939	0.939	0.939	0.939	0.944	0.930	0.999	0.955	0.999	0.996	0.989
6 planning	0.477	0.519	0.498	0.433	0.483	0.483	0.479	0.473	0.492	0.337	0.381	0.495	0.451
7 parkinsons	0.591	0.765	0.765	0.734	0.765	0.709	0.737	0.733	0.849	0.904	0.883	0.886	0.930
8 breast-cancer-wisc-prog	0.543	0.648	0.639	0.632	0.645	0.605	0.613	0.656	0.712	0.705	0.745	0.743	0.665
9 conn-bench-sonar-mines-rocks	0.675	0.686	0.696	0.676	0.691	0.685	0.701	0.699	0.868	0.834	0.887	0.834	0.876
10 glass-id	0.763	0.809	0.832	0.845	0.832	0.861	0.836	0.874	0.928	0.969	0.957	0.968	0.973
11 spect	0.574	0.655	0.648	0.667	0.638	0.634	0.634	0.642	0.663	0.737	0.740	0.707	0.754
12 spectf	0.515	0.711	0.770	0.718	0.773	0.723	0.732	0.763	0.799	0.811	0.837	0.785	0.792
13 statlog-heart-1	0.688	0.747	0.752	0.739	0.743	0.741	0.723	0.736	0.809	0.836	0.823	0.804	0.777
14 statlog-heart-2	0.812	0.829	0.833	0.837	0.839	0.838	0.803	0.838	0.900	0.866	0.907	0.905	0.902
15 breast-cancer-1	0.517	0.537	0.507	0.586	0.553	0.552	0.549	0.513	0.580	0.631	0.701	0.705	0.711
16 breast-cancer-2	0.487	0.554	0.551	0.601	0.545	0.588	0.574	0.577	0.687	0.659	0.665	0.711	0.732
17 breast-cancer-wang	0.528	0.553	0.579	0.586	0.576	0.571	0.598	0.578	0.645	0.603	0.683	0.671	0.598
18 colposcopies	0.545	0.653	0.639	0.663	0.639	0.677	0.649	0.660	0.726	0.678	0.734	0.763	0.715
19 heart-hungarian-1	0.739	0.766	0.788	0.756	0.791	0.769	0.771	0.782	0.825	0.841	0.834	0.872	0.860
20 heart-hungarian-2	0.758	0.773	0.798	0.749	0.793	0.762	0.790	0.746	0.889	0.878	0.908	0.904	0.912
21 breast-cancer-nki	0.512	0.578	0.556	0.585	0.572	0.588	0.582	0.572	0.752	0.644	0.713	0.752	0.711
22 horse-colic-1	0.604	0.735	0.708	0.717	0.705	0.703	0.715	0.695	0.678	0.698	0.724	0.698	0.789
23 horse-colic-2	0.744	0.793	0.793	0.790	0.796	0.806	0.763	0.789	0.873	0.777	0.877	0.881	0.881
24 haberman-survival	0.514	0.540	0.587	0.533	0.580	0.561	0.555	0.555	0.565	0.644	0.692	0.586	0.664
25 vertebral-column-2clases	0.779	0.820	0.820	0.808	0.818	0.823	0.776	0.816	0.877	0.921	0.906	0.931	0.911
26 mesothelioma	0.990	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
27 ionosphere	0.817	0.839	0.823	0.827	0.823	0.823	0.813	0.823	0.951	0.899	0.940	0.962	0.965
28 student-performance	0.510	0.594	0.585	0.640	0.600	0.598	0.615	0.613	0.722	0.651	0.695	0.715	0.682
29 congressional-voting	0.519	0.600	0.516	0.583	0.530	0.543	0.537	0.559	0.567	0.593	0.517	0.564	0.599
30 las-vegas-trip	0.496	0.551	0.559	0.504	0.543	0.486	0.491	0.510	0.657	0.650	0.638	0.644	0.663
31 cylinder-bands-1	0.547	0.545	0.565	0.583	0.563	0.566	0.539	0.588	0.670	0.681	0.687	0.692	0.737
32 cylinder-bands-2	0.555	0.624	0.636	0.616	0.635	0.616	0.616	0.637	0.811	0.750	0.808	0.833	0.803
33 breast-cancer-wisc-diag	0.956	0.977	0.969	0.972	0.969	0.962	0.969	0.967	0.986	0.986	0.996	0.987	0.988
34 lipd-indian-liver	0.536	0.572	0.546	0.577	0.563	0.586	0.610	0.536	0.500	0.739	0.710	0.500	0.720
35 climate-model-simulation-crashes	0.926	0.957	0.922	0.963	0.924	0.954	0.952	0.905	0.966	0.975	0.977	0.978	0.984
36 credit-approval-1	0.835	0.847	0.844	0.852	0.847	0.844	0.841	0.850	0.862	0.916	0.902	0.905	0.928
37 credit-approval-2	0.821	0.848	0.852	0.843	0.852	0.834	0.846	0.848	0.897	0.902	0.892	0.916	0.928
38 statlog-australian-credit-1	0.504	0.524	0.506	0.528	0.525	0.518	0.511	0.518	0.498	0.509	0.522	0.492	0.593
39 statlog-australian-credit-2	0.515	0.522	0.528	0.515	0.533	0.529	0.530	0.532	0.530	0.583	0.544	0.540	0.617
40 breast-cancer-wisc	0.958	0.962	0.963	0.956	0.962	0.968	0.957	0.962	0.982	0.979	0.992	0.988	0.989
41 cervical-cancer	0.529	0.594	0.589	0.611	0.574	0.606	0.609	0.566	0.600	0.655	0.591	0.653	0.658
42 pima	0.658	0.726	0.723	0.723	0.722	0.721	0.735	0.724	0.774	0.780	0.833	0.809	0.820
43 oocytes-trisopterus-nucleus-2f	0.690	0.697	0.694	0.695	0.693	0.686	0.693	0.686	0.816	0.831	0.851	0.866	0.741
44 tic-tac-toe	0.974	0.976	0.976	0.976	0.976	0.976	0.976	0.976	0.976	0.997	0.962	0.968	0.627
45 mammographic	0.773	0.812	0.811	0.816	0.812	0.812	0.812	0.811	0.829	0.787	0.767	0.835	0.851
46 statlog-german-credit-1	0.508	0.557	0.558	0.570	0.560	0.578	0.576	0.562	0.615	0.682	0.662	0.637	0.664
47 statlog-german-credit-2	0.538	0.666	0.663	0.652	0.667	0.668	0.650	0.677	0.777	0.750	0.779	0.786	---
48 oocytes-merluccius-nucleus-4d	0.709	0.718	0.715	0.731	0.726	0.726	0.730	0.729	0.683	0.850	0.832	0.781	0.743
49 hill-valley	---	---	---	---	---	---	---	---	0.730	0.752	0.695	0.777	0.525
Avg. AUC	0.677	0.722	0.720	0.725	0.723	0.723	0.717	0.721	0.778	0.785	0.795	0.798	0.793
Avg. rank	11.490	7.816	8.388	7.612	7.939	8.184	8.571	8.510	4.490	3.857	3.612	3.184	2.837

1st rank **2nd rank** 3rd rank

Fig. 4. **ICE outperforms the corresponding benchmark classifier on most datasets.** The ID-name mapping of datasets is shown in Figure 1. $L = 100$; $s = 0.5$; $w = 0.4$; $N = 5$; $\alpha = 1$; $\beta = 1$. ICE wins on 37 out of 49 datasets (%75.5).

Fig. 5. **Component analysis of ICE.** Each column indicates a randomized control experiment. **(a)** Marker '•' and '○' represent standard component and random control. **(b)** Color indicates the AUC gain of ICE over Bagging. **(c)** Each bar shows the average AUC gain of ICE over Bagging.

Bagging and ICE-Bagging) respectively. The base classifier is Bagging for the other six dynamic model selection methods - KNORA-U [17], KNORA-E [17], DES-PRC [18], [19], OLA [20], MCB [10] and A Priori [21], which is the suggested setting plus SVM to make comparable with other methods. We use the suggested parameters for dynamic model selection approaches [25].

As shown, all three versions of ICE have better performance than the other methods. The performance gain of ICE over Bagging can be attributed to the use of specifically generated models for subproblems and individualized model association and selection step. Comparing AdaBoost to ICE, both models attempt to produce subdomain-specific classifiers; however, AdaBoost always uses the same ensemble of all submodels for all instances, which reduces the potential performance gain provided by the submodel-specific models. Therefore, ICE-Adaboost and even ICE-SVM perform better than AdaBoost in general. More over, ICE outperforms the seven dynamic selection methods. Each of the dynamic selection methods has unique contributions on model selection or integration. However, none of them focuses on deliberately generating models for specific subproblems as the fuzzy clustering that ICE uses. In addition, the unique instance-model association of ICE can utilize all training instances, comparing to dynamic selection methods such as META-DES, which separates training data into META learning and dynamic selection datasets, therefore lead to more data loss and weaker base classifiers. As discussed earlier in Section 2E, learning the best combination of multiple randomly generated models can be a daunting task when the amount of training data is limited.
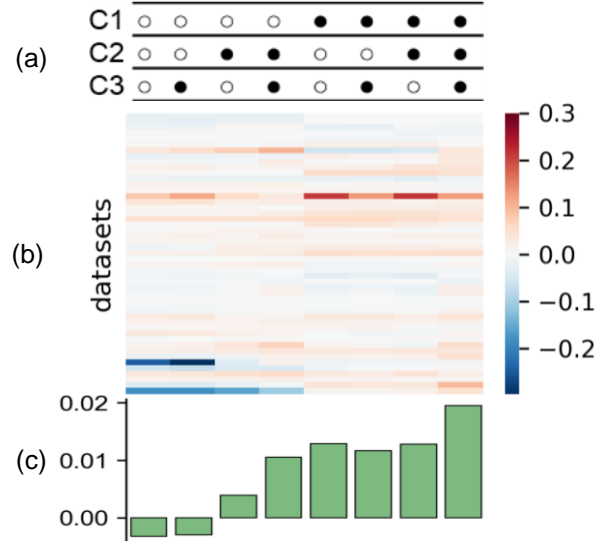
D. Randomized Control Analysis Reveals The Effectiveness of Different Components of ICE

To understand the impact of the three components of ICE (C1: fuzzy clustering based model generation, C2: instance-model association, and, C3: KNN-based model selection), we perform a randomized control experiment, where one or more of the components is replaced with comparable, randomized procedures. To randomize C1, the fuzzy clustering is replaced by bootstrapping instances , where the bags are made the same size as in the fuzzy clusters, therefore resulting in a slightly modified version of Bagging. To randomize C2, the decision table is shuffled row-wise, destroying the association of models to instances. Finally, to randomize C3, KNN is replaced with random selection of instances. Note that randomizing C2 or C3 (or both) are expected to have similar impact on the algorithm, which will essentially perform random model selection (and in most cases will choose many more models than real ICE due to independence of different rows of the randomized decision table).

Figure 5 shows the performance of ICE with different components randomized. Here, in order to show the effectiveness of each component of ICE, the parameter α and β are set to 0, effectively eliminating '*whole*' model. Not surprisingly, when both the model generation and model selection components of ICE are randomized (columns 1-3 in Figure 5a), its performance becomes similar to that of Bagging. On the other hand, when only one component is randomized (columns 4-7), ICE can still perform better than standard Bagging, although not as effective as the complete ICE algorithm (column 8), indicating that both components of ICE played a role in effective learning.
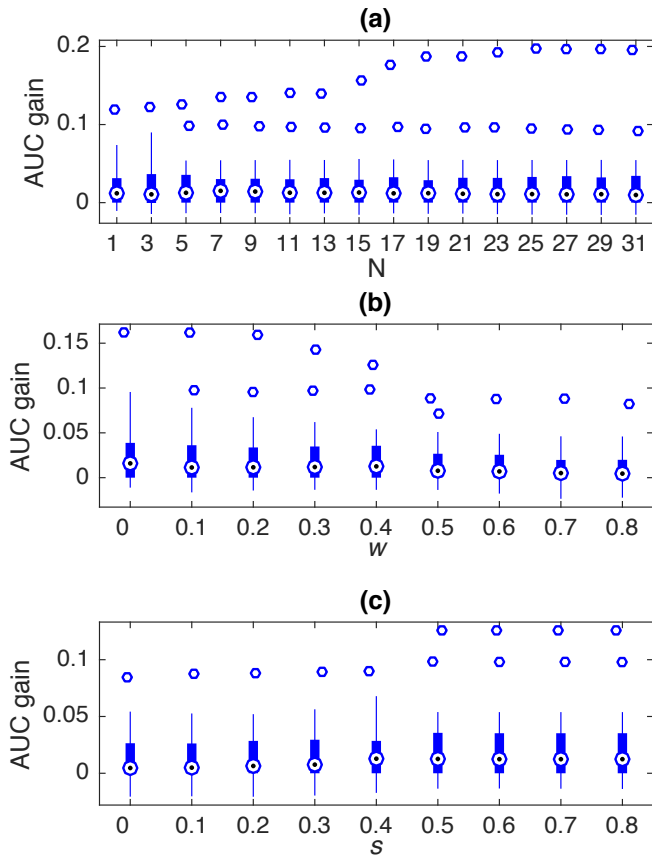
Fig. 6. **ICE performs in a stable manner across the wide range of parameter space.** **(a)** AUC gain varies as a function of N, number of nearest neighbors for model selection. Here $w = 0.4$, $s = 0.5$. **(b)** AUC gain varies as a function of w. Here $N = 5$, $s = 0.5$. **(c)** AUC gain varies as a function of s. Here $N = 5$, $w = 0.4$.

Interestingly, with only C1 randomized, our algorithm is conceptually similar to dynamic model selection [16], except that we replaced their learning-based model selection with simple KNN-based model selection. The fact that this version of ICE still outperforms dynamic model selection suggests that, with limited training data, KNN-based model selection can have more robust performance than learning-based model selection. In addition, when C2 or C3 (or both) are randomized but C1 is not randomized (column 5-7), our algorithm is conceptually similar to bagging, except that the models in the ensemble are based on clusters of instances instead of random selection of instances. As shown, this version of ICE has significant performance gain over Bagging, suggesting that, at least in these datasets, clustering-based model generation, which implicitly diversifies the models, can be better than randomized model generation.

E. Performance of ICE is Robust in a Wide Range of Parameter Space

Figure 6 shows the results of ICE using a wide range of parameters - N: number of neighbors per testing instances in prediction; w: the weight advantage of the base whole model

in model-instance association; s: the weight advantage of the self-model in model-instance association. In this analysis, the parameter α and β are both set to 1 to balance *whole* and *partial* models.

Figure 6a shows that the number of nearest neighbors used in model selection has only slight impact on AUC gain on average across all 49 datasets. The recommended setting of N is 5 to 10 for a balanced running speed and accuracy. ICE works best when there are strong patterns in the dataset. If ICE does not have a significant gain over Random Forest (RF) on a center dataset, a larger N setting will make ICE more stable and closer to bagging. ICE still has a large room of improvement on specific dataset by using more suitable fuzzy clustering algorithm, which is one of our future work.

Figure 6b and Figure 6c shows the robust performance of ICE with respect to parameter w and s. A general insight of w and s is to set s slightly larger than w, such as $s = 0.5$, $w=0.4$. The parameter α and β are quite simple to choose. Set both α and β to 1 will lead to a decent result for most of cases; try to set both α and β to 0 if there are strong clusters within the dataset, and the extreme localized classifiers may have an advantage over the basic to-go choice where $\alpha = \beta = 1$.

In addition, it is worth noting that the parameters used in the experimental setup have not been tuned for individual dataset in this study. There is a potential to perform model tuning on each dataset for even more improved performance.

F. ICE Significantly Improves Random Forest Performance

We further perform an extreme comparison between ICE (using Random Forest with 100 trees as the base classifier) and Random Forest with 10,000 trees. Random Forest (RF) is well known for its stable high performance with almost tuning-free design, and is well positioned to be a benchmark classifier. As shown in Figure 7, ICE significantly improves the performance of Random Forest; ICE wins or ties over RF on 36 out of 49 datasets (74%), and has minor performance loss on 13 datasets. The t-test p-value of gain = 0.018, which is significant (p-value< 0.02), and, there are 7 datasets (highlighted on Figure 7) with AUC gain over 8% (among these, ICE has AUC gain over 13.4% on 4 datasets), while no dataset with AUC loss over 3%. Note that ICE only uses on average 47 models per prediction, much fewer comparing to 10,000 trees by the RF classifier. Moreover, RF easily reaches its performance limits as the number of trees grows, while ICE has a much larger room of improvement as the number of submodels increases. Performance of ICE can be further improved by increasing the number of fuzzy clusters (submodels) or using more suitable clustering methods.

The performance gain of ICE over RF can be attributed to the use of specifically generated models for subproblems and individualized model association and selection step. Interestingly, the AUC gain of ICE is correlated with the result from Figure 3a - the 7 highest-scoring datasets by ICE on Figure 7 have on average 4.3 'partial' models winning the 'whole' model, while this statistic is only 2.6 for the other datasets; the average AUC gain by 'partial' models of ICE on these
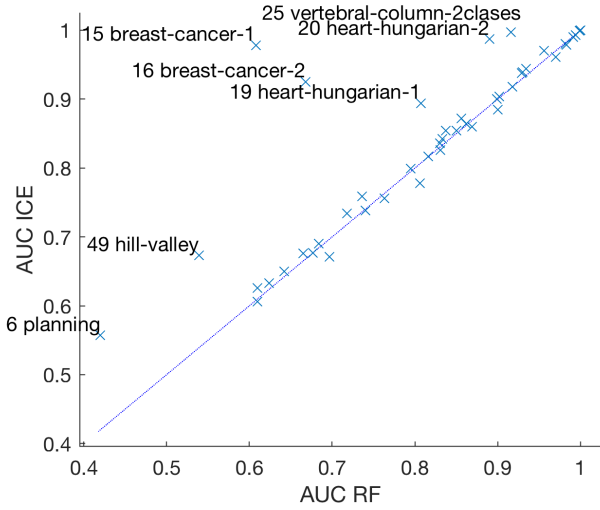
Fig. 7. **ICE significantly improves the performance of random forest on seven datasets.**
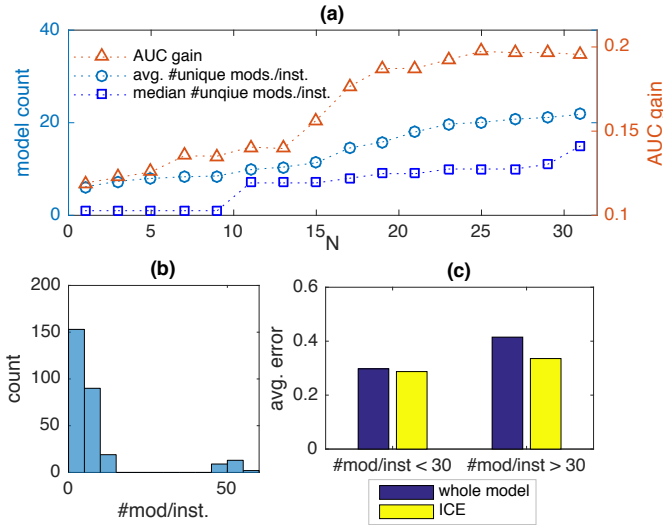

Fig. 8. **Analysis of ICE submodels on a breast cancer dataset. (a)** AUC gain of ICE varies as a function of model counts per instance. **(b)** distribution of number of unique '*partial*' models per instance. **(c)** Predictions by ICE have a lower error rate than Bagging on instances with >30 models.

7 datasets is 0.077, while it is 0.057 for the other datasets. This results not only further validates our intuition of using '*partial*' models to improve classification performance, but also suggests that the performance of ICE can be partially predictable based on dataset characteristics, which is a very important feature in practice.

G. Classification Improved by Accurately Predicting 'Hard' Instances with 'partial' Models

ICE has a stable AUC gain on most of datasets over a large range of parameter variation, and the dataset with one of the most dramatic improvement using ICE is the 15-breast-cancer-1 dataset. As shown in Figure 8a, as the parameter
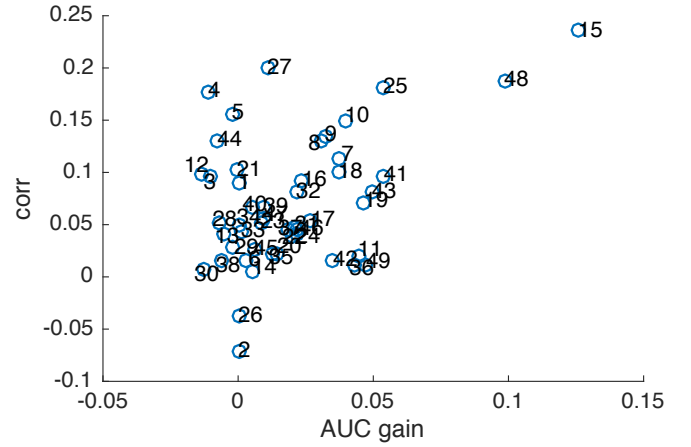

Fig. 9. **The higher the correlation between the data similarity and the decision table similarity, the higher the AUC gain.**

N increases, the average number of models per instance also increases and the performance of ICE continues to increase, reaching a plateau after $N \geq 25$. In addition, analysis of the models used by each test instance of ICE shows an interesting bimodal distribution: most of the test instances (262 out of 286 cases) use less than 20 models (mostly *local* models); in contrast, a few instances (24 cases) use more than 40 unique models (including both *local* and *remote* models) (Figure 8b), which are presumably the more difficult instances that are hard to be clustered and/or classified.

Comparing the performance difference on these two groups of instances, we can see that ICE has a much lower prediction error when compared to the '*whole*' model on instances with >30 models by ICE (Figure 8c). The *t*test *p*-value of the error differences between the '*whole*' model prediction and ICE prediction on instances with >30 models (24 cases) is significant (8.98×10^{-6}). This result demonstrates that different instances should be treated differently on this dataset, and the ICE algorithm shows a potential way of separating and treating these different instances.

H. AUC Gain of ICE has a Strong Correlation with the Data-Decision Table Similarity

In this work, instances are clustered based on their similarities in the feature space. However, it is possible that this clustering may not be optimal in revealing model heterogeneity. A different view may be obtained by analyzing the instance-instance similarities in the model space. Therefore, we use the decision table, which describes the prediction performance of each model on each instance, to measure instance-instance similarity, and inspect whether the consistency between these two types of similarity measures can be predictive of the performance of ICE.

Indeed, as shown in Figure 9, there exists a strong positive correlation (Pearson correlation coefficient = 0.425) between AUC gain and feature-model consistency, where the consistency is defined as the Pearson correlation correlation between

the instance-instance similarities measured in the feature space and the instance-instance similarities measured using the decision table entries. This result indicates the potential of improving our current work by feature selection and better clustering method on data X. Our intuition is that some features are more related with a classification task than the other features, and, we should be able to use these features for clustering for the classification task rather than use all the features. This also explains that the AUC gain on dataset 15-breast-cancer-1 (3 features, AUC gain = 0.126) is much larger than the AUC gain on dataset 16-breast-cancer-2 (50 features, AUC gain = 0.024). The three features of dataset 15-breast-cancer-1 are 'tumor-size', 'left or right breast' and 'if irradiate', and, all the non-binary nominal features in the original breast cancer dataset from [24] has been removed, while the dataset 16-breast-cancer-2 keeps all the other nominal features by One-Hot encoding. It is reasonable to imagine that the clustering on dataset 16 is influenced by some of the over-complicated and irrelevant features (for the classification task); therefore, the models built on those clusters are not optimized for the classification task. A potential future improvement is to cluster instances based on the output values from different models instead of on the feature values, or using both in an iterative manner.

IV. CONCLUSION

Based on the intuition that classifiers generated from different subdomains of training instances are needed in classification task, we proposed ICE, a novel multiple classifier generation and combination framework, which generally increases the diversity among submodels, and successfully associates the submodels to subdomains of instances. Evaluation results on 49 benchmarks show that our model has a stable improvement on a significant proportion of datasets over multiple existing MCS methods. A detailed component analysis shows that the different components of our algorithm work coordinately to achieve its performance. We believe that ICE can provide a novel choice of utilizing subdomain models to improve classification.

REFERENCES

[1] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[2] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[3] N. C. Oza and K. Tumer, "Classifier ensembles: Select real-world applications," *Information Fusion*, vol. 9, no. 1, pp. 4–20, 2008.

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

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

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

[7] R. Vilalta, M.-K. Achari, and C. F. Eick, "Class decomposition via clustering: a new framework for low-variance classifiers," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 673–676.

[8] L. I. Kuncheva, "Clustering-and-selection model for classifier combination," in *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, vol. 1. IEEE, 2000, pp. 185–188.

[9] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[10] G. Giacinto and F. Roli, "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognition*, vol. 34, no. 9, pp. 1879–1881, 2001.

[11] M. Gashler, C. Giraud-Carrier, and T. Martinez, "Decision tree ensemble: Small heterogeneous is better than large homogeneous," in *2008 Seventh International Conference on Machine Learning and Applications*, Dec 2008, pp. 900–905.

[12] M. J. Jahid, T. H. Huang, and J. Ruan, "A personalized committee classification approach to improving prediction of breast cancer metastasis," *Bioinformatics*, vol. 30, no. 13, pp. 1858–1866, 2014.

[13] H. Tong, C. Faloutsos, and J. y. Pan, "Fast random walk with restart and its applications," in *Sixth International Conference on Data Mining (ICDM'06)*, Dec 2006, pp. 613–622.

[14] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[15] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[16] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. I. Ren, "Meta-des: A dynamic ensemble selection framework using meta-learning," *Pattern recognition*, vol. 48, no. 5, pp. 1925–1935, 2015.

[17] A. H. Ko, R. Sabourin, and A. S. Britto Jr, "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognition*, vol. 41, no. 5, pp. 1718–1731, 2008.

[18] M. Kurzynski, M. Krysmann, P. Trajdos, and A. Wolczowski, "Multi-classifier system with hybrid learning applied to the control of bioprosthetic hand," *Computers in biology and medicine*, vol. 69, pp. 286–297, 2016.

[19] T. Woloszynski, M. Kurzynski, P. Podsiadlo, and G. W. Stachowiak, "A measure of competence based on random classification for dynamic ensemble selection," *Information Fusion*, vol. 13, no. 3, pp. 207–213, 2012.

[20] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[21] G. Giacinto and F. Roli, "Methods for dynamic classifier selection," in *Image Analysis and Processing, 1999. Proceedings. International Conference on*. IEEE, 1999, pp. 659–664.

[22] M. J. Van De Vijver, Y. D. He, L. J. Van't Veer, H. Dai, A. A. Hart, D. W. Voskuil, G. J. Schreiber, J. L. Peterse, C. Roberts, M. J. Marton *et al.*, "A gene-expression signature as a predictor of survival in breast cancer," *New England Journal of Medicine*, vol. 347, no. 25, pp. 1999–2009, 2002.

[23] Y. Wang, J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer-van Gelder, J. Yu *et al.*, "Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer," *The Lancet*, vol. 365, no. 9460, pp. 671–679, 2005.

[24] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems," *J. Mach. Learn. Res*, vol. 15, no. 1, pp. 3133–3181, 2014.

[25] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, pp. 195–216, 2018.