HETEROGENEOUS ENSEMBLES FOR THE MISSING FEATURE PROBLEM

Loris Nanni, [1] Sheryl Brahnam, [2] Nicola Lazzarini, [3] and Carlo Fantozzi[1]

[1]DIE, University of Padua, Via Gradenigo, 6 - 35131- Padova – Italy e-mail: {loris.nanni, fantozzi} @dei.unipd.it
[2]CIS, Missouri State University, 901 S. National, Springfield, MO 65804, USA e-mail: sbrahnam@ missouristate.edu
[3]University of Padua, Via Gradenigo, 6 - 35131- Padova – Italy e-mail: lazza87@gmail.com

ABSTRACT

Missing values are ubiquitous in real-world datasets. In this work, we show how to handle them with heterogeneous ensembles of classifiers that outperform state-of-the-art solutions. Several approaches are compared using several different datasets. Some state-of-the-art classifiers, e.g., SVM and RotBoost, are tested first and coupled with the Expectation-Maximization (EM) imputation method. The classifiers are then combined to build ensembles. Using the Wilcoxon signed-rank test (reject the null hypothesis, level of significance 0.05), we show that our best heterogeneous ensembles, obtained by combining a forest of decision trees (a method that does not require any dataset-specific tuning) with a cluster-based imputation method, outperforms two dataset-tuned solutions: a stand-alone SVM classifier and a random subspace of SVMs, both based on LibSVM, the most widely used SVM toolbox in the world. Our heterogeneous ensembles also exhibit better performance than a recent cluster-based imputation method for handling missing values (a method which has been shown to outperform several other state-of-the-art imputation approaches) when both the training set and the testing set contain 10% missing values. The MATLAB code of several tested descriptors, along with the datasets used in our experiments, is available at http://www.dei.unipd.it/wdyn/?IDsezione=3314&IDgruppo_pass=124& preview=.

Keywords: missing values; imputation methods; support vector machine; decision tree; ensemble of classifiers.

1. INTRODUCTION

Whether in business or in other domains, real data are rarely perfect. In most cases they are incomplete, vague, and inaccurate: survey questions are riddled with random or missing answers, images are often out of focus, sensors fail, and equipment malfunctions. These and other circumstances, such as extreme noise and accidental deletions and errors, can introduce values in datasets that bias estimation and prediction [24].

Of the various problems encountered in data sources, missing data is one of the most common [17]. Methods for handling missing values depend on the severity of the problem. In general, a missing data rate lower than 1% is not significant, while rates between 1% and 5% are considered manageable. Missing data rates between 5% and 15%, however, require sophisticated imputation

methods for treating the missing values, and rates greater than 15% are generally considered to have a severe impact on results. Missing values can be measured in the simplest case by searching for null values, but there are other forms of missing values that are more difficult to detect, such as outliers and values that that lie outside predetermined ranges [22].

Missing values are particularly problematic when working with machine learning algorithms. In neural networks and K-Nearest neighbor algorithms, for instance, missing values can result in variance underestimation, distribution distortion, and correlation depression. In some cases, the algorithms are unable to handle missing data. Unfortunately, the most common method for dealing with the problem in machine learning is essentially to ignore it [13] by using a technique commonly referred to as "filtering." This method for handling missing values works sufficiently well when missing data rates are below 5%, but becomes increasingly unsatisfactory when rates are higher. In some application areas, e.g., in epistatic miniarray profiling (E-MAP) [21], which is used for the analysis of genetic-interaction maps in the form of symmetric matrices, missing data rates can exceed 30%. In this case, discarding all genes that contain missing values would result in removing more than 90% of the data in the E-MAP matrix. Filtering in these cases greatly compromises the value of such studies.

In cases where missing values exceed 5%, there are a number of statistical models that could be used to assess the impact of missing values and to determine the best imputation method for treatment. According to Little and Rubin [17], models of missing data randomness can be divided into three classes: MCAR (**M**issing **C**ompletely **A**t **R**andom), where the probability for a random variable *X* is independent of the actual value of X or the values of the other features; 2) MAR (**M**issing **A**t **R**andom), where the missing probability is independent of the value of X after controlling the other features; and 3) NMAR (**N**ot **M**issing **A**t **R**andom), where the probability that X is missing might depend on the value of X itself. MCAR is usually the model of choice, but in many real-world applications MAR is a more realistic assumption [18].

Typical methods for data imputation essentially replace a missing value with one which has the greatest similarity to others in the dataset, as in *hot-deck* imputation [11], where new values for a feature are computed as the mean of that feature across the training set or across *k*-nearest neighbors. Robust statistical approaches have also been developed. A popular choice is multiple imputation (MI) [26], which replaces missing values by their $m > 1$ simulated versions (m typically small). Each of the simulated datasets is analyzed using standard methods, and the results are combined to produce estimates and confidence intervals that incorporate the uncertainty introduced by the missing data. For recent surveys on imputation methods, see [6] [23] [8].

The last ten years have witnessed the development of powerful machine learning imputation methods that fill in missing values by constructing a predictive model to estimate the values that are lacking from information in the dataset. These investigations have examined the application of some well-known stand-alone classifiers, such as the Multi-Layer Perceptron (MLP), K-nearest neighbors (KNN), self-organizing maps (SOM) and decision trees (DT) [28]. Bayesian networks have also produced some good results [9] [12]. In [12], for example, two Bayesian methods for imputation are proposed that are based on the construction of Bayesian networks for each feature with missing values.

A number of excellent studies of various imputation methods compared with or combined with machine learning algorithms have also recently appeared. In [28] single imputation, likelihood-based multiple imputation (MI), probabilistic split, and surrogate split are compared for coping with missing values in decision trees. Using real datasets, comparisons show MI to be the top performer. In [4], an iterative boosting method is proposed for improving the quality of the imputed features,

and, in [5], the same authors investigate the influence of imputation on classification error for five methods. These studies conclude that imputation is usually beneficial for the classification of instances with missing values. In [7], a comparison of imputation methods that includes some commercial methods for MI (such as Amelia II, WinMICE, and SAS) reports that only machine learning algorithms (the study considers three: MLP, KNN and SOM) significantly improves the classification results in the presence of missing values. Moreover, all three machine learning algorithms are shown to perform equally well.

A slightly different machine learning solution is proposed in [23], which investigates a novel random subspace (RS) approach dubbed Learn++.MF. This approach makes two assumptions: 1) the feature set is partially redundant and 2) the redundancy is distributed randomly over the feature set. Learn++.MF is based on the distribution update concepts of Learn++, with the random feature selection of RS. When an instance with missing values is introduced, only those classifiers trained with the features that are presently available in that test pattern are used for classification. In their study, the authors compare Learn++.MF with the one-class approach [14], the expectation-maximization (EM) approach [27], and another RS-based approach where the mean imputation is used for the missing values before the RS ensemble classification. They show that Learn++.MF outperforms these other methods across several datasets.

In this work, EM is used as the base imputation method, and we boost performance in the presence of missing values using a heterogeneous ensemble built on SVM classifiers as well as other ensembles. We compare our ensemble approach with other state-of-the-art classification methods using several datasets. Our best approach outperforms both a stand-alone SVM (even when the SVM kernel and the various SVM parameters are carefully fine-tuned for each dataset) and a recently-proposed ensemble [20], which has already been shown to outperform several other state-of-the-art methods for handling missing values.

2. PROPOSED APPROACH

The heterogeneous ensembles we propose are based on the fusion by sum rule [15] of different state-of-the-art classifiers, which in turn may be ensembles of classifiers themselves (e.g., a random subspace of RotBoost). In our opinion, the main value of such *ensembles of ensembles* is that they offer the most feasible way of coping with the "no free lunch" theorems, which state that any two optimization algorithms in a suitable class exhibit the same performance when results are averaged across all possible problem sets (see, e.g., [29]). In other words, heterogeneous ensembles may be one of the best performing methods across various classification problems in the presence of missing values. Different ensembles are tested in this paper. In the remainder of this section, we describe the building blocks composing these ensembles.

2.1 Expectation-Maximization (EM) [1]

EM [27], first proposed in 1977 [2], is an iterative method for parameter estimation that uses maximum likelihood criteria. In this work we use EM as the core method to fill in missing values before training the classifiers in our ensembles.

EM formalizes an intuitive method to manage missing values and can be outlined as follows:

1. Estimate initial parameters using complete data.
2. Use parameters to estimate and evaluate missing data.

[1] Details and MATLAB code: http://www.gps.caltech.edu/~tapio/imputation/ (accessed: 18 October 2012).

3. Use the *new* estimated data to reevaluate parameters.
4. Repeat steps 2 and 3 until convergence.

Formally, let a dataset, given as a data matrix $\mathbf{X} \in R^{n \times p}$ with n patterns and p features, contain some rows (patterns) with missing values. For each pattern x_i let a be the portion of the row for which values are available (p_a columns) and m the portion where values are missing ($p_m = p - p_a$ columns). Let us define $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as the mean and the covariance matrices of the dataset, which are correspondingly partitioned between available and missing values for a given row. The goal of EM is to replace the missing values with values that are consistent with the available data. The missing values are assumed to be randomly scattered according to a normal distribution.

For each row x_i the relation between the available and the missing portions is modeled by a linear regression:

$$\mathbf{x}_m - \boldsymbol{\mu}_m = (\mathbf{x}_a - \boldsymbol{\mu}_a)\mathbf{B} + e, \qquad \textbf{EQ. 1}$$

where e is the imputation error, with zero mean and unknown covariance matrix $\mathbf{C}$, and $\mathbf{B}$ is the regression coefficients matrix. EM estimates $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\mathbf{B}$, and $\mathbf{C}$ (for details see [27]). Using the estimated parameters, the missing values are filled in and new estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are calculated. The whole process is repeated until values for estimated data are stable.

2.2 Random Subspace (RS)

RS [10] is an ensemble combining technique that is used in some our ensemble experiments. Consider a training set $\mathbf{X}$ formed by n vectors $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ with p features each. RS randomly selects $r < p$ features from the original feature space and creates a new training set $\mathbf{X}^{\mathrm{r}}$. Each vector $\mathbf{x}_i \in \mathbf{X}^{\mathrm{r}}$ is thus r-dimensional. A classifier is built using the new training set $\mathbf{X}^{\mathrm{r}}$, i.e., using a reduced feature space. This process is repeated b times, so that b random feature subsets and b classifiers are created. The final classification is obtained by combining the scores of the b classifiers.

An outline of the RS algorithm is the following:

1. **For** i=1 **to** b **do:**
   a. Select an r-dimensional random subspace $\mathbf{X}^{\mathrm{r}}$ from the original p-dimensional feature space
   b. Train a classifier $C_i(x)$ on $\mathbf{X}^r$

   Endfor
2. Combine classifiers $C_i(x)\ i = 1, \ldots, b$ by a simple combination method. The sum rule is used in this paper:

$$score_{RSM} = \frac{1}{b}\sum_i score_{C_i(x)} \qquad \textbf{EQ. 2}$$

2.3 Support Vector Machine (SVM)

SVM [1] is a linear binary classifier. In our work we use SVMs as core classifiers in several of our ensembles. An SVM performs classification by cutting the n-dimensional space, with n being the number of features, into two regions associated with two distinct classes which are often referred to as the *positive class* and the *negative class*. The regions are separated by an n-dimensional hyperplane that has the largest possible distance d from the training vectors of the two classes. To

formally define an SVM, consider a set of training vectors $(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)\ldots$, where $x_i \in R^n$ denotes the *i*-th input vector and $y_i \in \{+1, -1\}$ is the corresponding label. If there are unequal misclassification costs, the optimization problem becomes:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|w\|^2 + C_+ \sum_{i:y_i=1} \xi_i + C_- \sum_{j:y_j=-1} \xi_j \\ &\text{Subject to } y_k(w \cdot x_k + b) \geq 1 - \xi_k, \quad \xi_k \geq 0, \end{aligned} \qquad \textbf{EQ. 3}$$

where *w* is a vector normal to the hyperplane, $|b|/\|w\|$ is the distance of the hyperplane from the origin, $\|w\|$ is the Euclidean norm of *w*, C_+/C_- are classification costs that enable a tradeoff between training errors (C_+ for positive training patterns and C_- for the negative patterns) and the margin *d*. The slack variable ξ_i allows for errors in classification.

To handle classes that are not linearly separable, kernel functions are used that remap the training vectors into a higher *k*-dimensional space (*k* >> *n*) where examples become separable. With the inclusion of kernel functions and Lagrange multipliers α_i, the dual optimization problem can be formulated as:

$$\begin{aligned} &\text{maximize } w(\alpha) \sum_{i=1}^{l} \alpha_i - \sum_{i=1;j=1}^{l} \alpha_i y_i \alpha_j y_j K(x_i \cdot x_j) \\ &C \geq \alpha_i \geq 0 \ \ \forall i \ \ \textstyle\sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \qquad \textbf{EQ. 4}$$

Training examples with a nonzero α are called Support Vectors (SVs). The hyperplane is completely defined by the SVs, which denote the training points lying at minimum distance from the hyperplane itself. The solution of the classification problem becomes a decision function:

$$f(x) = sign\left(\sum_{i=1}^{N_{SV}} \alpha_i y_i K(s_i, x) + b\right), \qquad \textbf{EQ. 5}$$

where *x* is a vector to be classified, the s_i's are the support vectors (N_{SV} in total), and $K(s_i, x)$ is the kernel function.

2.4 Random subspace of RotBoost with NPE (RSR)

Rotation Boosting [30], or RotBoost, combines two ensemble techniques: AdaBoost and Rotation Forest. Experimental results reported in [30] show that RotBoost outperforms several variants of AdaBoost and Rotation Forest.

Starting from the original code shared by the authors of [30], in this paper the RotBoost method is improved by combining it with RS. Moreover, the Rotation Forest part of the method adopts the neighborhood preserving embedding (NPE) feature transform in lieu of PCA. Our choice is motivated by [19], which shows that RotBoost coupled with NPE outperforms standard RotBoost. RotBoost and NPE are described in detail below.

2.4.1 Rotation Boosting (RotBoost)

RotBoost [30] is an ensemble classifier technique obtained from a combination of AdaBoost and Rotation Forest. AdaBoost [32] is a sequential ensemble algorithm where a new classifier is built

each iteration by taking into account the performance obtained by the classifier created in the previous iteration. A weight is associated with each training pattern. At the beginning of an iteration, the weights of patterns that were misclassified by the previous classifier are increased, and the weights of correctly classified instances are decreased. In this way, the method focuses on difficult instances, since in each step the goal is to correctly classify instances that were misclassified in the previous iteration.

RotationForest [31] is an ensemble technique that builds each classifier on a training set that is modified by applying PCA. The primary heuristic is to exploit a feature transform method and then reconstruct a full feature set for each classifier included in the ensemble. This is performed in the following way: the feature set is randomly split into K subsets, or parameters, and PCA is applied separately to each subset. By pooling all principal components, a new set of *n* features is obtained. In this way data are linearly mapped into the new feature space, and classifiers are trained using them. The goal is to promote diversity through feature extraction while preserving accuracy by keeping all the extracted principal components.

The RotBoost algorithm is a simple combination of AdaBoost and Rotation Forest. PCA is applied first as in RotationForest, and a Rotation Matrix is obtained that maps data into a new feature space. Base classifiers are then built by applying the AdaBoost technique. As demonstrated in [30], RotBoost exhibits significantly lower misclassification errors with respect to both AdaBoost and RotationForest.

2.4.2 Neighborhood Preserving Embedding (NPE)

Neighborhood Preserving Embedding (NPE) [33] is an algorithm that solves the general problem of dimensionality reduction.[2] Given a set of points $x_1, x_2, \dots, x_m \in R^m$, the idea is to find a transformation matrix **A** that maps these points into another set $y_1, y_2, \dots y_m \in R^d$ where $d \ll m$. In this way, $y_i = A^T x_i$ represents x_i in a space with significantly less dimensions.

NPE begins by building a weight matrix to describe the relationships between data points: each point is described as a weighted combination of its neighbors. An optimal embedding is sought such that the neighborhood structure is preserved in the reduced space.

The algorithm can be formalized in three steps:

1) *Build an adjacency graph.* Define a graph **G** with *m* nodes. The *i*-th node represents the point x_i. There is an edge between *i* and *j* iff x_j is one of the K nearest neighbors of x_i;
2) *Compute weights.* In this step weights on edges are calculated. **W** is the weight matrix and W_{ij} is the weight of the edge from node *i* to node *j*. The matrix can be computed by minimizing the objective function:

$$\min \sum_i^m \left\| x_i - \sum_j^m W_{ij} x_j \right\|^2 \qquad \text{EQ. 6}$$
$$\text{Subject to: } \sum_j^m W_{ij} = 1, j = 1,2, \dots, m;$$

3) *Compute the Projection.* In this step the linear projection is computed. The following eigenvector problem is solved: $XMX^T\mathbf{a} = \boldsymbol{\lambda} XX^T\mathbf{a}$**.** The local manifold structure is then preserved using the following transformation matrix **A** that maps x_i to y_i:

[2] MATLAB code available from http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html.

$$y_i = A^T x_i \text{ , where } \mathbf{A} = (\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_{d-1}). \quad \textbf{EQ. 7}$$

2.5 Cluster-Based Imputation (CBor_EM)

Cluster-based imputation (*CBor_EM*) [20] is an ensemble built using a multiple imputation approach. First the missing features are calculated using EM, then the training patterns are clustered into *CN* groups (*CN*=5, in this paper). For each group a separate EM imputation is performed, and the resulting training set is used to build a classifier, which is a random subspace of SVMs. Finally, the *CN* classifiers are combined by sum rule. Moreover, in the original algorithm the multiple imputation method is coupled, by sum rule, with a random subspace of SVMs trained using the whole dataset and EM (based on the whole dataset) as imputation method (see [20] for details). The experiments in [20] show that the multiple imputation outperforms other state-of-the-art imputation approaches (CBor_EM is compared with a number of recent methods using more than 10 datasets).
In this paper we test the original method based on EM using the original parameters (i.e., the same number of imputations and parameters of the clustering procedure) without any fine-tuning. We also examine a variant of the method where random subspaces are built before the clustering and imputation step. This method is referred to as *CBI* in the experimental section. This variant should work better with correlated features since random subspace is performed before the other steps.

Pseudocode for CBI can be written as follows:

TRAINING

1. NORMALIZATION
   1.1 Normalize the original training patterns (OTR) and the testing patterns to the 0÷1 interval. Replace the missing values in the training set using the EM imputation, thus obtaining a new training set called ETR.

For *k* = 1 to *50* **do**:

2. RANDOM SUBSPACE
   Extract from OTR and ETR two random subspaces, OTR_k and ETR_k, that retain 50% of the set of original features
3. CLUSTERING
   3.1 Use the fuzzy-based clustering method to cluster the patterns of ETR_k into *CN* groups (we simply set *CN* =10)

   For *t* = 1 **to** *CN* **do**:

   3.2 Let IDX be the indices of training patterns in ETR_k that have a similarity to the *t*-th cluster greater than *TH* (we fix *TH*=0.25 as in [20]). Let D_t be a subset of OTR_k built using the patterns whose indices belong to IDX

   3.3 **While** D_t contains less than 25 patterns **or** there exists a feature whose value is missing in all the patterns that belong to D_t:
   assign to D_t a random subset of 25% of all the training patterns
4. EM IMPUTATION
   4.1 Fill in the missing values of OTR_k by performing a new EM imputation that uses only the training patterns in D_t. Let $ETR_{k,t}$ be the training set built using M_t
5. CLASSIFICATION
   5.1 Train a support vector machine using $ETR_{k,t}$ and classify the test patterns

   Endfor

Endfor

6. FUSION
   6.1 Combine the scores obtained by the different SVMs by sum rule

3. EXPERIMENTAL RESULTS

To compare our system with other state-of-the-art approaches, we report results obtained using the *Datasets with induced missing values* from the KEEL Repository.[3] Table 1 shows the main characteristics of the datasets in terms of the number of features, examples, and classes. A detailed description of these databases is available on the UCI machine learning website.[4]

In the KEEL datasets only the training partitions are affected by missing values: to be precise, the datasets have been built by randomly removing 10% of the values. We have also modified the datasets to introduce 10% or 25% of missing values in the test partitions for performing further tests. The features in each dataset are linearly normalized to the 0÷1 interval before using them to train the classifiers.

DATASET	Abbreviation	#Features	#Examples	#Classes
Iris	IR	4	150	3
Pima	PI	8	768	2
Wine	WI	13	178	3
Australian	AU	14	690	2
Newthyroid	NT	5	215	3
Ecoli	EC	7	336	8
German	GE	20	1000	2
Magic	MA	10	1902	2
Shuttle	SH	9	2175	7
Satimage	SA	36	6435	7

Table 1. Characteristics of the datasets used in the experiments: number of features, number of examples, and number of classes.

A 10-fold cross-validation protocol is used in all experiments. For comparison purposes, it is possible to download the 10-fold cross-validation partitions from the KEEL Repository.

The error area under the ROC curve (EUC) [3, 25] is used as the performance metric. The ROC curve is a plot of the sensitivity versus false positives (1 - specificity). The error area under the ROC curve can be interpreted as the probability that the classifier will assign a lower score to a randomly chosen positive sample than it would to a randomly chosen negative sample. When a multiclass dataset is used, the one-versus-all area under ROC curve is used as the performance indicator [16]. In this case, the final EUC value is obtained by averaging all class values. The area under the ROC curve is considered one of the most reliable performance metrics [25] since it is based on both sensitivity and specificity.

Table 2 shows the results of the following classifiers when the original KEEL datasets (where only the training sets contain missing values) are used:

- **SVM**: stand-alone SVM (see Section 2.3) with kernel and parameters tuned separately for each dataset.
- **RSS**: random subspace of SVMs (see Section 2.2).
- **RSR**: random subspace of RotBoost with NPE (see Section 2.4).
- **CBor_EM**: cluster-based imputation (see Section 2.5) as proposed in [20]. The number of subspaces is set to 50.

[3] http://sci2s.ugr.es/keel/missing.php#sub2b (accessed: 8 October 2012).

[4] http://archive.ics.uci.edu/ml/ (accessed: 8 October 2012).

- **CBI**: variant of cluster-based imputation (see Section 2.5). The number of subspaces is also 50.
- **CBI+RSS**: CBI coupled with RSS.
- **SVM+RSR**: sum rule between SVM and RSR.
- ***X*SVM+RSR**: weighted sum rule between SVM and RSR. The weight of SVM is *X* while that of RSR is one.
- ***X*CBor+RSR**: weighted sum rule between CBor_EM and RSR. The weight of SVM is *X* while that of RSR is one.

The implementation of SVM used in all our experiments is the one provided by the popular LibSVM library.

The outcome of the comparison is summarized in Table 2. The column named *Av* is the average rank and reports the relative position of a classifier against the others (if a given classifier is consistently the best in all datasets, then its rank would be 1).

DATASET	AU	EC	GE	IR	MA	NT	PI	SA	SH	WI	Av
SVM	6.94	3.89	24.68	0.20	15.92	0.12	17.05	1.09	0.49	0.04	7.9
RSS	7.37	4.17	23.04	0.53	15.45	0.22	17.00	0.94	0.69	0.04	9.1
RSR	7.02	4.65	21.26	0.27	12.77	0.10	17.27	1.00	**0.01**	**0.00**	6.0
CBor_EM	7.31	4.13	22.29	0.27	14.02	0.12	16.69	0.94	0.06	0.04	7.7
CBI	7.66	5.21	23.14	1.93	16.40	3.38	18.47	1.18	1.16	**0.00**	10.9
CBI+RSS	7.46	4.34	22.93	0.93	15.57	0.48	17.46	1.05	0.84	**0.00**	9.6
1SVM+RSR	6.81	3.88	21.81	**0.13**	13.11	**0.06**	16.50	0.83	0.03	**0.00**	**3.0**
2SVM+RSR	6.81	3.81	22.55	**0.13**	13.48	**0.06**	16.55	0.86	0.06	0.04	4.8
3SVM+RSR	**6.75**	**3.71**	22.96	**0.13**	13.82	0.15	16.59	0.89	0.09	0.04	6.1
1CBor +RSR	6.81	4.07	**20.53**	0.20	**11.75**	**0.06**	16.06	**0.80**	0.02	**0.00**	3.1
2CBor +RSR	6.94	4.01	20.84	0.27	11.86	**0.06**	15.97	0.83	0.03	**0.00**	4.3
3CBor +RSR	7.06	4.09	21.06	0.20	11.99	**0.06**	**15.90**	0.86	0.04	0.04	5.5

Table 2. EUC obtained for the different datasets and classification methods under consideration. Best results for each dataset are marked in bold.

Several interesting observations can be made from the reported results:

- A random subspace of SVMs (RSS) does not outperform a stand-alone SVM. This can be explained by observing that the datasets have few features; hence, on average, there is no correlation problem. RSS works best when there is either the curse of dimensionality and/or a correlation problem [23]. Probably for this reason the performance of CBI and CBI+RSS is lower than that obtained by the original CBor_EM method proposed in [20].
- The best results are obtained by the sum rule between SVM and RSR (i.e., 1SVM+RSR and 1CBor +RSR). In our opinion, it is very interesting that our heterogeneous ensembles outperform the most widely used SVM library (LibSVM) even when SVM is finely tuned for each dataset. As our ensembles work significantly better than LibSVM, we are making our MATLAB code publicly available for others to use.

The differences in performance between 1SVM+RSR (or 1CBor+RSR), the highest-ranking ensemble, and SVM, RSS, and CBor_EM are all statistically significant, as demonstrated by applying the Wilcoxon signed-rank test [3]. In each of these cases, the *p*-value of rejecting the null hypothesis (which assumes 1SVM+RSR exhibit the same EUC as the other classifier) was found to be less than 0.05. While the performance of 1SVM+RSR and 1CBor+RSR are similar.

The datasets in Table 1 have also been used in the literature (see [18]) for the evaluation of a different set of classifiers that operate with missing values. In [18], accuracy was selected as the measure of performance. In Table 3, we compare the accuracy of our best ensemble with the results reported for the same datasets in [18]. In both our results and in [18], the same 10-fold cross-validation partitions are adopted. It is clear that our approach exhibits superior performance across the line.

DATASET	AU	EC	GE	IR	MA	NT	PI	SH	WI
SVM+RSR	**87.68**	**88.40**	**76.50**	**96.67**	**81.81**	**96.30**	**77.23**	**99.45**	**98.30**
[18]	85.71	67.70	72.26	95.73	77.13	92.13	72.49	97.57	96.52

Table 3. Comparison using accuracy as the performance metric with the methods in [18].

DATASET	AU	EC	GE	IR	MA	NT	PI	SA	SH	WI	Av
SVM	10.96	5.37	24.61	**0.67**	16.55	1.49	16.72	1.15	0.37	0.13	8.3
RSS	9.17	6.45	23.29	0.73	15.37	1.62	16.70	0.98	0.21	0.08	8.0
RSR	**7.69**	6.51	**21.38**	0.80	13.02	0.70	18.58	1.03	0.02	0.09	6.1
CBor_EM	8.77	6.05	23.11	**0.67**	14.21	0.83	16.98	**0.80**	0.11	0.05	6.5
CBI	7.98	8.13	23.99	2.27	21.30	2.70	19.49	1.23	4.47	0.17	10.8
CBI+RSS	7.98	6.67	24.44	1.40	17.97	0.74	17.75	1.08	1.57	0.08	9.5
1SVM+RSR	8.49	**5.37**	21.83	**0.67**	13.42	0.58	16.86	0.87	0.04	**0.04**	**3.9**
2SVM+RSR	8.85	6.45	22.63	0.73	14.02	0.58	16.65	0.92	0.04	**0.04**	5.4
3SVM+RSR	9.25	6.51	23.09	0.73	14.47	0.62	**16.51**	0.95	0.11	**0.04**	6.9
1CBor +RSR	8.06	5.68	21.57	0.73	**12.95**	**0.45**	17.00	0.82	**0.02**	**0.04**	**3.9**
2CBor +RSR	8.19	5.83	21.77	**0.67**	13.15	0.48	16.88	0.79	0.04	**0.04**	4.0
3CBor +RSR	8.29	5.91	21.99	**0.67**	13.24	0.58	16.71	0.77	0.05	**0.04**	4.7

Table 4. EUC obtained using the datasets where both training/testing partitions contain 10% of missing values. Best results for each dataset are marked in bold.

DATASET	AU	EC	GE	IR	MA	NT	PI	SA	SH	WI	Av
SVM	10.69	6.46	26.57	1.53	19.01	4.96	18.60	1.31	0.69	0.27	8.5
RSS	10.16	7.22	26.07	2.33	18.49	4.80	18.69	1.07	0.43	0.24	8.6
RSR	8.68	6.35	**23.24**	2.33	**15.47**	2.36	19.74	1.15	0.42	0.40	6.2
CBor_EM	9.01	6.97	25.65	1.60	17.40	2.65	18.52	0.86	0.35	**0.13**	5.7
1SVM+RSR	9.11	**5.36**	23.91	1.67	15.71	2.24	18.25	0.96	0.18	0.28	4.3
2SVM+RSR	9.45	5.40	24.61	1.60	16.18	2.94	18.38	1.02	0.25	0.29	5.8
3SVM+RSR	9.76	5.53	25.01	**1.40**	16.58	3.36	18.41	1.05	0.30	0.22	5.7
1CBor +RSR	**8.46**	6.14	23.30	1.67	15.56	**1.83**	18.30	0.90	**0.14**	0.24	3.2
2CBor +RSR	8.55	6.37	23.55	1.53	15.78	1.89	**18.07**	0.86	0.18	0.19	**3.0**
3CBor +RSR	8.69	6.40	23.80	1.67	15.99	2.01	18.15	**0.85**	0.20	0.15	4.0

Table 5. EUC obtained using the datasets where both training/testing partitions contain 25% of missing values. Best results for each dataset are marked in bold.

As a last experiment, we compare the classifiers using our modified datasets that introduce 10% and 25% missing values both in the training patterns and in the testing patterns. In table 4 we report the results obtained using the datasets with 10% of missing values, while in Table 5 we report the results with 25% of missing values.

In these tests, both 1SVM+RSR and 1CBor +RSR do not perform better than CBor_EM with p-value < 0.05 but with p < 0.10. However, considering that in the previous test 1SVM+RSR/1CBor+RSR outperformed CBor_EM, we can conclude that they are more consistent in providing a high level of performance. To corroborate this observation, further tests should be performed with different percentages of missing values. We also note that both 1SVM+RSR and 1CBor+RSR continue to perform better than the two SVM-based approaches (i.e., SVM and RSS), as confirmed by the Wilcoxon test: the *p*-value of rejecting the null hypothesis is, again, found to be less than 0.05.

Finally, we remark that the additional tests with an increased number of missing values do not allow us to discriminate the performance of the two best ensembles, 1SVM+RSR and 1CBor+RSR. With 10% of missing values in the test set, the two ensembles obtain the same performance; with 25% of missing values, 1CBor+RSR outperforms 1SVM+RSR but with too high a p-value (i.e., 0.25) for a solid conclusion. All in all, our experiments in this paper point out 1CBor+RSR is the best performing ensemble. However, further datasets should definitely be considered to assess the performance difference, if any, between 1CBor+RSR and 1SVM+RSR.

4. CONCLUSION

In this paper we experimentally compare several classifiers and heterogeneous ensembles of classifiers to assess their performance in the presence of missing values. Ten different datasets are considered, addressing a wide spectrum of diverse problems. We test three versions of the datasets. In the first version, only the training data contains missing values (10%). In the second and third versions, both the training and testing patterns contain missing values (10%/25%, respectively).

We identify an *ensemble of ensembles*, which we call *1CBor+RSR*, that outperforms two prominent solutions:

1. The most widely used SVM library (LibSVM), even when SVM is finely tuned for each dataset. We remark that our result is not trivial since, as we showed in Section 3, a simple ensemble of SVMs is not enough to boost performance.
2. The original method for missing imputation proposed in [27], which in turn has already been shown to exhibit better performance than several state-of-the-art imputation approaches.

The superior performance of our ensemble is confirmed by the Wilcoxon signed-rank test.

In future work, we plan to investigate further the validity of our heterogeneous ensembles by applying them to a wider selection of datasets exhibiting several different characteristics of patterns.

ACKNOWLEDGEMENTS

The work described in this paper was done while Nicola Lazzarini was at the University of Padova. Support for some of the authors was provided in part by MIUR of Italy as part of the project *AlgoDEEP* and by the University of Padova as part of the project STPD08JA32 *AACSE*.

REFERENCES

[1] Cristianini, N., and Shawe-Taylor, J., *An introduction to support vector machines and other kernel-based learning methods*, Cambridge, UK: Cambridge University Press, 2000.

[2] Dempster, A. P., Laird, N. M., and Rubin, D. B., "Maximum likelihood from Incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[3] Demšar, J., "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7 pp. 1-30, 2006.

[4] Farhangfar, A., Kurgan, L., and Dy, J., "Impact of imputation of missing values on classification error for discrete data," *Pattern Recognition*, vol. 41, no. 12, pp. 3692-3705, 2008.

[5] Farhangfar, A., Kurgan, L. A., and Pedrycz, W., "A novel framework for imputation of missing values in databases," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 37, no. 5, pp. 692-709, 2007.

[6] Garcia-Laencina, P., Sancho-Gomez, J., and Figueiras-Vidal, A., "Pattern classification with missing data: a review," *Neural Computation & Applications of Soft Computing*, vol. 9, no. 1, pp. 1-12, 2009.

[7] Ghannad-Rezaie, M., Soltanian-Zadeh, H., Ying, H., and Dong, M., "Selection-fusion approach for classification of data sets with missing values," *Pattern Recognition*, vol. 43, pp. 2340-2350, 2010.

[8] Gheyas, I., and Smith, L. S., "A neural network-based framework for the reconstruction of incomplete datasets," *Neurocomputing*, vol. 73, no. 16-18, pp. 3039-3065, 2010.

[9] Hautaniemi, S., Egren, H., Vesanen, P., Wolf, M., Järvinen, A. K., Yli-Harja, O., Astola, J., Kallioniemi, O., and Monni, O., "A novel strategy for microarray quality control using Bayesian networks," *Bioinformatics*, vol. 19, no. 16, pp. 2031–2038., 2003.

[10] Ho, T. K., "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832-844, 1998.

[11] Howell, D. C., "The treatment of missing data," *The SAGE Handbook of Social Science Methodology*, William Turner and Stephen P. Turner, eds., pp. 208–224, London, UK: Sage, 2007.

[12] Hruschka Jr., E. R., Hruschka, E. R., and Ebecken, N. F., "Bayesian networks for imputation in classification problems," *Journal of intelligent information systems*, vol. 29, no. 3, pp. 231-252, 2007.

[13] Jerez, J. M., Molina, I., García-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., and Franco, L., "Missing data imputation using statistical and machine learning methods in a real breast cancer problem," *Artificial Intelligence in Medicine,* vol. 50, no. 2, pp. 105-115, 2010.
[14] Juszczak, P., and Duin, R. P. W., "Combining one-class classifiers to classify missing data," in Proceedings of the International Workshop on Multiple Classifier Systems (MCS 2004), Cagliari, Italy, 2004, pp. 92–101.
[15] Kittler, J., "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 3, pp. 226–239, 1998.
[16] Landgrebe, T. C. W., and Duin, R. P. W., "Approximating the multiclass ROC by pairwise analysis," *Pattern Recognition Letters,* vol. 28, pp. 1747–1758, 2007.
[17] Little, R. J., and Rubin, D. B., *Statistical analysis with missing data,* New York: John Wiley and Sons, 1987.
[18] Luengo, J., García, S., and Herrera, F., "A study on the use of imputation methods for experimentation with Radial Basis Function Network classifiers handling missing attribute values: The good synergy between RBFNs and EventCovering method," *Neural Networks,* vol. 23, no. 3, pp. 406-418, 2010.
[19] Nanni, L., Brahnam, S., Lumini, A., and Barrier, T., "Data mining based on intelligent systems for decision support systems in healthcare," *Intelligent Support Systems in Healthcare Using Intelligent Systems and Agents,* Sheryl Brahnam and Lakhmi C. Jain, eds.: Springer, 2010.
[20] Nanni, L., Lumini, A., and Brahnam, S., "An classifier ensemble approach for the missing feature problem," *Artificial Intelligence in Medicine,* vol. 55, no. 1, pp. 37-50, 2012.
[21] Pan, X. Y., Tian, Y., Huang, Y., and Shen, H. B., "Towards better accuracy for missing value estimation of epistatic miniarray profiling data by a novel ensemble approach," *Genomics,* vol. 97, no. 5, pp. 257-264, 2011.
[22] Pearson, R. K., *Mining imperfect data*, Philadelphia, PA: SIAM, 2005.
[23] Polikar, R., DePasquale, J., Syed Mohammed, H., Brown, G., and Kuncheva, L. I., "Learn++.MF : A Random Subspace Approach for the Missing Feature Problem," *Pattern Recognition,* vol. 43, no. 1, pp. 3817-3832, 2010.
[24] Pyle, D., *Data preparation for data mining,* San Francisco, CA: Morgan Kaufmann Publishers, 1999.
[25] Qin, Z. C., "ROC analysis for predictions made by probabilistic classifiers," in Fourth International Conference on Machine Learning and Cybernetics, 2006, pp. 3119-312.
[26] Rubin, D. B., *Multiple imputation for nonresponse in surveys,* New York: J. Wiley & Sons, 1987.
[27] Schneider, T., "Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values," *Journal of Climate,* vol. 14, pp. 853–871, 2001.
[28] Twala, B., "An empirical comparison of techniques for handling incomplete data using decision trees," *Applied Artificial Intelligence,* vol. 23, pp. 373-405, 2009.
[29] Wolpert, D. H., "The supervised learning no-free-lunch theorems," in 6th Online World Conference on Soft Computing in Industrial Applications, 2001, pp. 25-42.
[30] Zhang, C.-X., and Zhang, J.-S., "RotBoost: a technique for combining Rotation Forest and AdaBoost,," *Pattern Recognition Letters,* vol. 29, no. 10, pp. 1524-1536, 2008.