Using Imputation Techniques to Help Learn Accurate Classifiers

Xiaoyuan Su
Computer Science & Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
xsu@fau.edu

Taghi M. Khoshgoftaar
Computer Science & Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
taghi@cse.fau.edu

Russell Greiner
Computing Science
University of Alberta
Edmonton, AB, Canada
greiner@cs.ualberta.ca

Abstract

It is difficult to learn good classifiers when training data is missing attribute values. Conventional techniques for dealing with such omissions, such as mean imputation, generally do not significantly improve the performance of the resulting classifier. We proposed imputation-helped classifiers, which use accurate imputation techniques, such as Bayesian multiple imputation (BMI), predictive mean matching (PMM), and Expectation Maximization (EM), as preprocessors for conventional machine learning algorithms. Our empirical results show that EM-helped and BMI-helped classifiers work effectively when the data is "missing completely at random", generally improving predictive performance over most of the original machine learned classifiers we investigated.

1. Introduction

Missing data is unfortunately common in data analysis; this typically leads to difficulties in estimation and inference. To deal with missing data, many analytic tools either ignore the missing values or fill in the missing values with a value estimated by some process -- this process is called *imputation*.

As the predictive performance of many learning algorithms deteriorates on incomplete training data, some try using imputation techniques to suggest values for missing data before training a classifier. However, many such machine learning systems use a simple imputer. Some use *mean imputation* (MEI), which replaces the missing value with the mean value of the attribute over all instances or over all instances of the same class, or with the most frequently observed value of the attribute (e.g., this is used by the WEKA [20] implementation of logistic regression [4] and random forest [3]). Others fill in the missing value with a global constant, such as the value "unknown" (e.g., WEKA's sequential minimal optimization [14] and one rule classifier [7]). Yet others simply ignore the missing values (e.g., WEKA's naïve Bayes [9] and decision table [11]). However, empirical results show that these simple imputation methods typically do not significantly improve the machine learners' performance.

We are therefore motivated to propose imputation-helped classifiers, which use better imputation techniques for filling in the missing values before training the classifiers. To our knowledge, no one has empirically compared various state-of-the-art imputation techniques, such as Expectation Maximization (EM) [5] (Section 2.4), predictive mean matching (PMM) [12] (Section 2.5), and Bayesian multiple imputation (BMI) [16] (Section 2.6), to see which one(s) best help standard machine learning algorithms to deal with incomplete data. This work explores this task.

We considered datasets from the UCI machine learning repository [2] whose values were artificially deleted using the MCAR (missing completely at random) mechanism (defined below), using different pre-determined missing ratios. After using imputation techniques to impute missing values, we learn classifiers on this complete data using the following 10 machine learning algorithms: decision tree (C4.5), decision table (dTable), k nearest neighbor (kNN), logistic regression (LR), naïve Bayes (NB), neural networks (NN), one rule (OneR), decision list (PART), support vector machine (SVM), and random forest (RF). We then evaluate the predictive performance improvement of these various imputation-helped classifiers over the original classifiers.

Section 2 presents the framework of our work and reviews relevant literature; then Section 3 provides experimental design and results.

2. Framework

The imputation-helped classifiers work in the following steps (see Figure 1): impute values for the incomplete training data to generate imputed (complete) training data; learn a classifier on the imputed dataset;

then have the classifier produce classification for the test dataset.

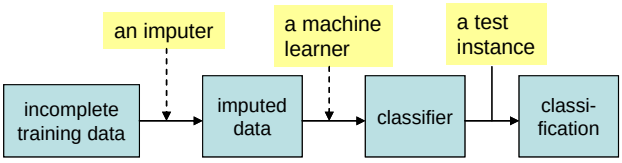

Figure 1. Framework of imputation-helped classifiers

2.1 Patterns of Missing Data

Little and Rubin [13] consider three types of missing data patterns. If the missingness does not depend on the observed data, then it is regarded as Missing Completely at Random (*MCAR*). This includes the situation where a value is missing with an iid (independent and identical distribution) probability value (e.g., 0.2) that is independent of the actual value of this feature, or of any other feature (think of random corruption while the data is being transmitted). Simple imputation techniques, such as mean imputation and linear regression imputation, are often used to deal with such MCAR missing data.

This work will focus on MCAR missingness. To help explain this, we contrast it with two other types of missingness. Missing data that depends on the observed data, but not on the unobserved data is considered Missing at Random (*MAR*) [17]. For example, if a person's gender is recorded as male, then the attribute "pregnancy" is typically left blank. Full maximum likelihood and Bayesian imputation techniques can be used to deal with missing data that are *MAR* as well as *MCAR* [10].

Both *MCAR* and *MAR* missing data patterns are considered *ignorable* patterns, as valid inferences and estimations can be made even if the missing data pattern is not explicitly considered in the analysis.

If the "missingness" is not *MAR*, then we say the data are Not Missing at Random (*NMAR*, or *non-ignorable*). Here, patterns of the unobserved variables can determine their own missingness. For example, the "years of education" field of a job application may be more likely to be omitted if it is "< 1". Incorrectly assuming ignorability can result in biased estimates [17]. When the missing data are *non-ignorable*, a model of missing value mechanism must be incorporated into the estimation process to produce meaningful estimates [19]. This is generally specific to the problem itself and is difficult in practice, especially as the missing data mechanism is rarely known with certainty. The good news is that multiple imputation [16], which works well for both MAR and MCAR data, can often reduce the bias of the estimates when data are NMAR.

2.2 How Machine Learners Handle Missing Data

When we use a machine learning algorithm, we usually do not know the missing patterns of the data beforehand. We therefore want a learner that works well for any pattern of missingness. In this work, we consider 10 well-known machine learning algorithms from *WEKA* [20], each of which has its own strategy to deal with missing data. (Note that none of these approaches attempts to determine the type of missingness.)

2.2.1 Ignore Missing Values. *Decision table (dTable)* classifier includes two components: a schema that is a set of features, and a body consisting of labeled instances from the space defined by the features in the schema [11]. Given an unlabelled instance, the *dTable* classifier searches for exact matches in the table only using the features in the schema. If it finds no matching instances, it then returns the majority class; otherwise, it returns the majority class of all matching instances. *DTable* ignores missing values during the learning and classification processes.

Naïve Bayes (NB) is an extremely simple Bayesian network that assumes attribute values are conditionally independent given the class, and typically assumes that numeric attributes obey a Gaussian distribution [9]. *NB* learns by estimating the conditional distributions of each attribute given the class; here it simply ignores attribute values that are missing.

Decision tree (C4.5) is a member of the *ID3* family of algorithms that grows decision trees from the root downward, greedily selecting the next attribute for each new decision branch added to the tree. In the learning stage, *C4.5* algorithm just ignores missing values in gain and entropy calculation, and in the classification stage, it assigns a probability to each of the possible values of the missing value based on the number of training instances going down that branch, divided by the total number of training instances with observed values at the node [15].

Decision list (PART) is a decision list classifier based on partial decision trees. It combines *C4.5* and *RIPPER* to avoid their respective problems to produce accurate rule sets. *PART* deals with missing values using the same strategy as *C4.5* [6].

2.2.2 Use Mean or Median of Observed Values. The *logistic regression (LR)* implementation in *WEKA* uses a multinomial logistic regression model with a ridge estimator, and uses a *ReplaceMissingValuesFilter* to replace the missing values with the mean (for numeric attributes) or the most frequent value (for nominal attributes) [4].

Random Forest (*RF*) grows many classification trees. To classify a new instance, it first asks each tree in the forest for its classification, and then returns the

classification having the most votes. An *RF* learner replaces the missing values with the median value for numeric attributes or the most frequent value for nominal attributes [3]. *RF* fills in the missing data in test set using filled-in values from the training set

2.2.3 Missing Values Replaced by a Default Value. *One rule (OneR)* is a rule-based classifier that infers one rule that predicts the class, based on the most informative single attribute. The attributes are assumed to be discrete, otherwise they must be discretized. Missing values are treated as the new value, "missing" [7].

A kernel-based *Support vector machine (SVM)* produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space. WEKA's *SVM* uses the sequential minimal optimization (*SMO*) algorithm [14] for training a support vector classifier using polynomial (which we used) or *RBF* kernels. This implementation globally replaces all missing values by a default value, e.g., "unknown".

2.2.4 Missing Values Used in Distance Measure. A *k-nearest-neighbor* (*kNN*) classifier finds the *k* neighbors that are closest to the unknown instance, and returns the average value of the real-valued labels of the neighbors. The closeness of the neighbors is defined in terms of *Euclidean* distance for continuous attributes and *Hamming* distance for discrete ones. *KNN* handles missing values by means of a minor change in the distance measure: when the two instances each miss the values of the same attribute, the distance on that attribute is zero, but when only one has a missing value, a maximal distance is assigned [20].

2.2.5 Other Methods to Deal with Missing Values. A *Neural network (NN)* is composed of interconnected input/output units, where each connection has an associated weight, learned using the backpropagation algorithm. Many neural network models have been modified to handle missing data. Following [8], we replace each missing value by an interval that includes all the possible values on that attribute (eg., a unit interval [0,1]), and also replace each observed value by a degenerate interval (e.g, 0.7 transformed to [0.7,0.7]), before applying the backpropagation algorithm. Learning and making classification from incomplete data are therefore turned to classification of the (complete) interval vectors.

2.2.6 Summary. Each of the above machine learning algorithms uses a simple approach to deal with missing data in its learning process, regardless of the missing patterns. They may perform poorly on incomplete data, especially those missing many values. This paper explores the use of more advanced imputation techniques, such as BMI, EM, or PMM (introduced below). As some of our imputers only deal with numerical or ordinal datasets (but not nominal), we only investigate these kinds of data.

2.3 Linear Regression Imputation

Linear regression (*LinR*) imputation attempts to predict the missing value based on the observed values of other variables. In general, given a one-dimensional vector of inputs $\boldsymbol{X}=(X_1, X_2, \ldots, X_p)$, linear regression predicts the dependent value Y based on the linear regression model

$$Y = \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \varepsilon \quad (1)$$

where ε is a residual and coefficients β_0 and $\boldsymbol{\beta}=(\beta_1, \beta_2, \ldots, \beta_p)^T$ are trained on the existing values to minimize the sum of squared residuals. Here Y is the missing feature value to be imputed, and each X_j is the value of an observed feature of the same instance.

We round *LinR* imputed values to the nearest integers for integer attributes. We also find the observed value range [*min*, *max*] for each attribute, and replace imputed values <*min* with *min*, and those >*max* with *max* for missing values. This procedure is also applied for other imputers described below.

2.4 Expectation Maximization Imputation

Expectation-maximization (EM) is a well-known algorithm that seeks maximum likelihood estimates of parameters in probabilistic models in the presence of latent variables [5]. EM imputation requires specifying a joint probability distribution for the feature value to be imputed and the other feature values. EM iterates between performing an expectation E step, which computes an expected value of the complete data likelihood, given the observed data and the current parameters; and a maximization M step, which calculates values of the parameters that maximize the expected likelihood over the data, including those estimated in the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated until it converges to a stationary point.

Our implementation of EM algorithm assumes the data is drawn from a multivariate Gaussian data (parameterized by the mean and the covariance matrix) and uses ridge regression [18]. EM first produces an initial guess of these parameters. In each subsequent iteration, EM updates its estimates of the mean and the covariance matrix in three steps [18]: (1) for each instance with missing values, initialize the distribution parameters by estimating the mean and the covariance matrix; (2) the missing values in a sample are replaced with their conditional expectation values given the observed values using the estimated mean and covariance

matrix; (3) the mean and the covariance matrix are re-estimated, using the sample mean of the completed dataset and the covariance matrix as the sum of the sample covariance matrix of the completed dataset and the contributions from the conditional covariance matrix of the imputation errors. EM iterates these steps until the imputed values and the estimates of the mean and covariance stop changing [13].

2.5 Predictive Mean Matching Imputation

Predictive mean matching (*PMM*) is a state-of-the-art imputation technique [12] that imputes missing values $Y_{miss,i}$ of incomplete instance (recipient) Y_i, based on the observed part of that instance $Y_{obs,i}$ to find the nearest instance (donor) using a distance function (see Equation 3 below) that is computed as the expected values of the missing variables conditioned on the observed covariates, instead of directly on the values of the covariates. Our version of *PMM* works as follows:

1) Use the *EM* algorithm [5] to estimate the parameters θ of a multivariate Gaussian distribution over the attribute values using all the available data.

2) Compute the conditional expected value for the missing part $Y_{miss,i}$ of instance Y_i conditioned on the observed part $Y_{obs,i}$ based on the estimated parameters θ.

$$\hat{\mu}_i = E(Y_{miss,i} \mid Y_{obs,i}, \theta) \tag{2}$$

3) Match each recipient Y_i to another instance (possible donor) Y_j=*argmin*$_j$ *d(i,j)* that has the nearest predictive mean with respect to the *Mahalanobis distance,* defined through the residual covariance matrix from the regression of the missing items on the observed ones.

$$d(i,j) = \sqrt{(\hat{\mu}_i - \hat{\mu}_j)^T \, S^{-1}_{Y_{miss,i}|Y_{obs,i}} (\hat{\mu}_i - \hat{\mu}_j)} \tag{3}$$

where the S is the empirical covariance matrix [21].

4) Impute each missing value in the recipient with the corresponding values from its closest donor.

2.6 Bayesian Multiple Imputation

Standard *single imputation* produces a single filled-in dataset, where each missing value is replaced with a single value, perhaps drawn from the posterior predictive distribution. While this approach is simple and can be applied to virtually every data set, single imputation does not account for the uncertainty about the predictions of the imputed values, which can lead to statistically invalid inferences [16]. By contrast, *multiple imputation* (MI) produces many different filled-in datasets. For example, consider filling in the (3,3) position in the upper-left 4x3 table in Figure 2. Here, we could produce m=4 different completed datasets, shown as the next 4 tables in that figure; note the proposed values for this (3, 3) entry are {2,3,4,3}. BMI uses the average of these four values (here, 3) as the final prediction – see the bottom right 4x3 table in the figure. In many situations, *MI* approaches have proven to be highly effective even for small values of m – say 3 to 12 [16].

BMI follows a Bayesian framework: it specifies a parametric model for the complete data, with a prior distribution over the unknown model parameters θ, then simulates m independent draws from the conditional distribution of the missing data given the observed data by Bayes' Theorem. We apply Markov chain Monte Carlo (*MCMC*) to perform *BMI* [16].

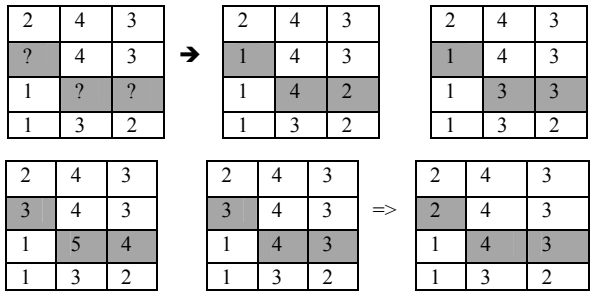

Figure 2. An example of BMI with *m=4*. Each value in the shaded cells is an estimated value from an imputation. BMI produces different filled-in datasets (after the "➔" sign) and takes the average of the *m* predictions as the final imputed dataset (after the "=>" sign).

While *BMI* assumes a multivariate normal distribution when generating the imputations for missing values, it is robust to non-normally distributed data [17]. *BMI* imputes data as follows [16]:

Let $P(Y_{com}|\theta)$ model the complete data, based on the parameter θ (which here is the mean and covariance matrix that parameterize a normal distribution here). If $Y= (Y_{obs}, Y_{miss})$ follows a parametric model $P(Y|\theta)$ where θ has the prior distribution $P(\theta)$, then the posterior predictive distribution for Y_{miss} is

$$P(Y_{miss} \mid Y_{obs}) = \int P(Y_{miss} \mid Y_{obs}, \theta) P(\theta \mid Y_{obs}) d\theta \tag{4}$$

Equation 4 suggests that *BMI* can be drawn by iterating the following process for $j=1, \ldots, m$:

(1) generate missing values $Y_{miss}^{(j+1)}$ from $P(Y_{miss}|Y_{obs}, \theta^{(j)})$;

(2) draw parameters from $\theta^{(j+1)}$ from $P(\theta|Y_{obs}, Y_{miss}^{(j+1)})$.

Repeat these two steps to generate the Markov chain $\{Y_{miss}^{(1)}, \theta^{(1)}, Y_{miss}^{(2)}, \theta^{(2)}, \ldots, Y_{miss}^{(j)}, \theta^{(j)}, \ldots\}$; note that $Y_{miss}^{(j+1)}$ depends on $\theta^{(j)}$, and $\theta^{(j)}$ depends on $Y_{miss}^{(j)}$. This entire process is repeated until the distribution $P(Y_{miss}, \theta|Y_{obs})$ is stabilized [17].

After producing m sets of filled-in values, we take the average as the final imputed set of values,

$$\overline{Q} = \frac{1}{m} \sum_{i=1}^{m} \hat{Q}_i \tag{5}$$

where $\hat{Q}_i$ is the i-th imputed value.

3. Experimental Design and Results

We used eight datasets with numeric or ordinal attributes from the UCI machine learning repository [2] (see Table 1). We ran all 10 classifiers on each dataset; in general, we used the default parameters given by WEKA. For each, we used 50 trees for the random forest classifier (*RF50*), and used k=5 as the number of neighbors for the *kNN* classifier; we found these settings produced optimal performance in our preliminary experiments. When implementing our proposed imputation-helped classifiers, we use an iteration number of 5 for LinR and PMM, and BMI and EM iterate until they converge. When preprocessing the incomplete data with the imputers, we impute the training and test sets together. Notice none of the imputers use the class labels of the test sets.

We used the standard training and test splits for each of these datasets: 2/3 of the instances for training and 1/3 for testing, except the dataset "letter", which was a 3/4 to 1/4 split. We take the average classification accuracy of 5 runs. In each run, with MCAR missing pattern, we generate three incomplete datasets from each dataset by randomly deleting 10%, 30%, and 50% of the observed values.

Table 1. Description of the UCI datasets we worked on

datasets	# Train	# Test	# attri	# class
australian	460	230	14	2
breast	466	233	10	2
diabetes	512	256	8	2
heart	180	90	13	2
pima	512	256	8	2
shuttle-small	3866	1934	9	7
vehicle	564	282	18	4
letter	15000	5000	16	26

We computed the average classification accuracy over the 8 datasets for the original classifiers, and MEI-helped, PMM-helped, LinR-helped, EM-helped, and BMI-helped classifiers: 10% missing appears in Table 2 and Figure 3; 30% missing in Table 3 and Figure 4; and 50% missing in Table 4 and Figure 5. We did not count the results on the dataset "letter" for the classifier kNN as it produces exceptionally low accuracy (<10%) from the original classifier when the dataset has missing ratios of 30% and 50%. Our results collectively show that EM-helped classifiers perform the best: they achieve the highest average classification accuracy -- significantly higher than the original classifiers (those without using imputers) with 1-sided t-test p-value $p<0.00037$. BMI-helped classifiers are significantly better than original classifiers with $p<0.0029$. MEI-helped classifiers (resp., LinR-helped, and PMM-helped ones) however, do not significantly outperform original classifiers, with p-values of $p<0.18$ (resp. $p<0.40$, and $p<0.12$).

Table 2. Average classification accuracy over eight datasets with MCAR missing ratio 10%

	original classifier	MEI - helped	LinR-helped	EM-helped	PMM-helped	BMI-helped
OneR	68.31	67.98	69.18	69.59	68.78	**69.85**
NB	**75.01**	74.60	74.02	74.62	74.11	74.74
dTable	76.29	76.29	78.36	**79.16**	76.85	78.38
C4.5	**81.77**	80.01	80.26	81.01	79.51	81.28
PART	**82.51**	80.17	80.98	81.50	79.73	80.91
SVM	80.68	80.75	81.66	**82.50**	80.81	82.20
LR	80.23	80.14	82.24	**83.03**	80.82	82.62
NN	79.00	80.35	79.72	82.19	80.21	**82.21**
RF50	**84.88**	83.97	83.90	84.69	83.71	84.69
kNN	73.75	81.07	82.13	83.16	81.49	**83.38**
ave	78.24	78.53	79.24	**80.15**	78.60	80.03

Table 3. Average classification accuracy over eight datasets with MCAR missing ratio 30%

	original classifier	MEI - helped	LinR-helped	EM-helped	PMM-helped	BMI-helped
OneR	64.19	63.68	65.50	**69.31**	65.65	68.66
NB	**73.34**	71.93	70.98	72.95	70.48	72.31
dTable	71.42	72.19	70.50	**75.40**	71.33	74.30
C4.5	76.59	75.22	71.97	**77.37**	72.04	76.05
PART	**77.65**	74.66	72.20	77.10	72.44	76.87
SVM	75.06	75.00	73.79	**78.63**	73.95	77.92
LR	75.58	75.33	72.70	**77.85**	73.93	76.91
NN	72.40	74.65	73.70	**78.17**	73.35	78.00
RF50	80.48	79.68	77.29	**80.78**	76.89	80.19
kNN	62.19	77.43	76.91	**80.71**	77.21	80.66
ave	72.89	73.98	72.55	**76.83**	72.71	76.19

Table 4. Average classification accuracy over eight datasets with MCAR missing ratio 50%

	original classifier	MEI - helped	LinR-helped	EM-helped	PMM-helped	BMI-helped
OneR	61.48	60.86	64.19	**67.60**	63.48	66.54
NB	**71.29**	68.80	66.88	70.48	67.38	69.94
dTable	67.34	67.91	66.91	**72.10**	66.71	70.39
C4.5	71.05	69.60	66.41	**73.82**	65.87	71.85
PART	72.48	70.40	67.31	**72.65**	65.79	71.76
SVM	70.84	71.19	69.71	**73.72**	67.92	72.90
LR	71.63	71.73	68.88	**73.65**	67.70	71.48
NN	68.13	70.40	67.76	**73.86**	66.55	72.96
RF50	76.33	73.82	70.70	**76.52**	69.76	75.34
kNN	58.14	72.23	72.80	**77.42**	71.74	77.38
ave	68.87	69.69	68.15	**73.18**	67.29	72.05

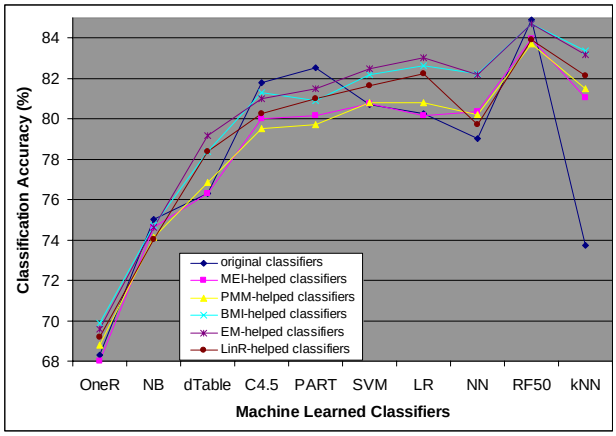

Figure 3. Average classification accuracy over eight datasets with MCAR missing ratio 10%

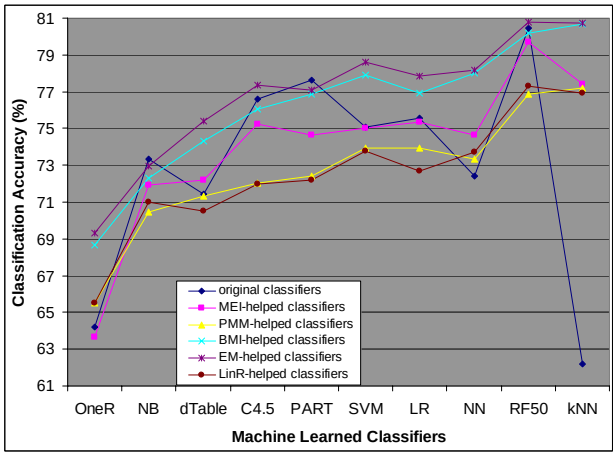

Figure 4. Average classification accuracy over eight datasets with MCAR missing ratio 30%

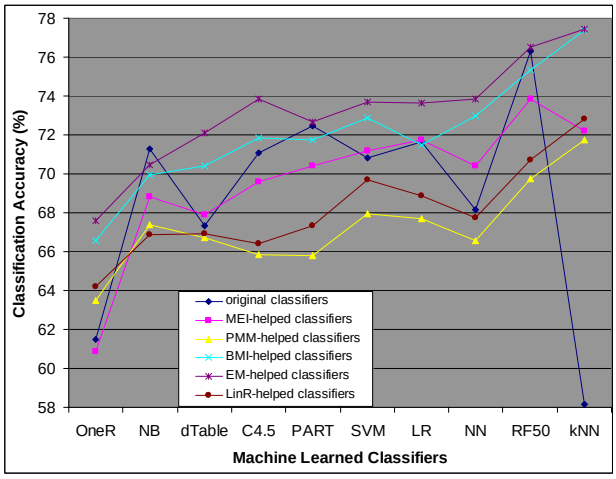

Figure 5. Average classification accuracy over eight datasets with MCAR missing ratio 50%

When we look into individual classifiers and examine their performance with different imputers, especially the best performed EM-helped and BMI-helped classifiers, for incomplete datasets with different missing ratios, we observe the following (see Table 5, Figure 6, and Table 6).

(1) Our imputation-helped classifiers best help kNN, as the average classification accuracy improvement of the five *imputation-helped classifiers with kNN* over the original kNN classifier is 11.5% on the datasets with 10% missing rario; 26.4% improvement on the 30% missing data; and 27.8% improvement on the 50% missing data.

(2) EM-helped and BMI-helped classifiers perform significantly better than the original classifiers *neural network*, *one rule*, *decision table*, and *SVM*, for missing data with all ratios we investigated in this work.

(3) The EM-helped classifier improves the *linear regression* classifier for missing data with all ratios we investigated.

(4) EM-helped classifiers slightly improve the *decision tree* (C4.5), and *random forest* classifiers especially for incomplete data with high missing ratios, i.e., 30% and 50%, and it also slightly improves the PART classifiers on incomplete data with 50% missing ratio.

(5) Imputation-helped classifiers techniques generally do not help improve the *naïve Bayes* classifier.

(6) LinR-helped and PMM-helped classifiers can slightly improve the *neural network*, *one rule*, *decision table*, and *SVM* classifiers (as well as kNN) when the missing ratio is 10%; however, when the missing ratio is increased to 30%, they can only slightly improve the *neural network*, and *one rule* classifiers; and when missing ratio is 50%, only the *one rule* classifier (see Table 6).

Table 5. Average accuracy improvement of BMI-helped and EM-helped classifiers over original classifiers on missing data with ratios 10%, 30%, and 50%

	10%		30%		50%	
	BMI	EM	BMI	EM	BMI	EM
kNN	13.06	12.76	29.70	29.78	33.09	33.16
NN	4.065	4.032	7.726	7.966	7.078	8.404
OneR	2.264	1.883	6.960	7.978	8.239	9.953
dTable	2.736	3.762	4.038	5.576	4.522	7.057
SVM	1.878	2.250	3.817	4.761	2.906	4.066
LR	2.974	3.491	1.765	3.007	-0.21	2.813
C4.5	-0.60	-0.92	-0.71	1.025	1.124	3.899
RF50	-0.23	-0.22	-0.35	0.371	-1.30	0.246
PART	-1.93	-1.22	-1.00	-0.71	-1.00	0.231
NB	-0.36	-0.53	-1.41	-0.54	-1.89	-1.14

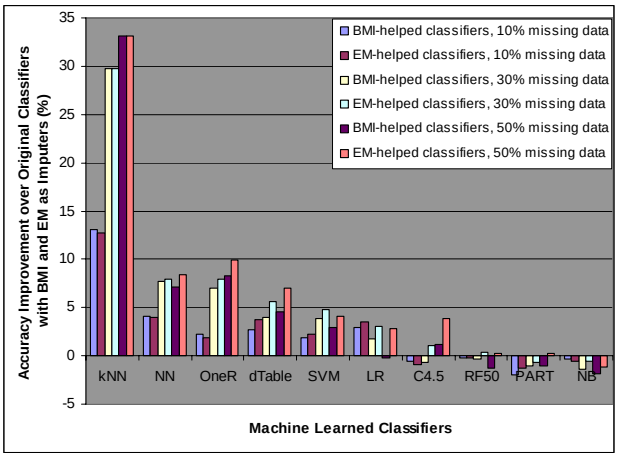

Figure 6. Average accuracy improvement of BMI-helped and EM-helped classifiers over original classifiers on missing data with ratios 10%, 30%, and 50%

Table 6. Average accuracy improvement of LinR-helped and PMM-helped classifiers over original classifiers on missing data with ratios 10%, 30%, and 50%

	10%		30%		50%	
	LinR	PMM	LinR	PMM	LinR	PMM
kNN	13.79	5.633	23.67	24.15	25.21	23.39
NN	0.905	1.533	1.797	1.305	-0.55	-2.33
OneR	1.279	0.692	2.035	2.086	4.408	3.249
dTable	2.702	0.724	-1.30	-0.12	-0.64	-9.34
SVM	1.218	0.164	-1.69	-1.48	-1.60	-4.11
LR	2.501	0.728	-3.81	-2.18	-3.83	-5.48
C4.5	-1.84	-2.76	-6.03	-5.94	-6.53	-7.30
RF50	-1.16	-1.38	-3.97	-4.46	-7.37	-8.61
PART	-1.85	-3.36	-7.01	-6.71	-7.13	-9.24
NB	-1.32	-1.20	-3.22	-3.91	-6.19	-5.48

(7) Besides kNN, the MEI-helped classifiers can also improve the *neural network* classifier for datasets with all the missing ratios we investigated.

The above results are the *average* predictive performances of the imputation-helped classifiers on the eight datasets. However, they conceal the fact that, while an original classifier (without using imputers) can, *on average*, outperform BMI-helped or other imputation-helped classifiers (this is true for *NB* for datasets with missing ratios of 10% through 50%), this is not true for every single dataset. Figure 7 is a learning curve of different imputation-helped classifiers with NB as the machine learner on the dataset "australian", with missing ratios from 10% through 90%, in which the *BMI-helped classifier with NB* almost dominates over the others (except a better performed imputer EM), and it performs especially well for datasets with high missing ratios.

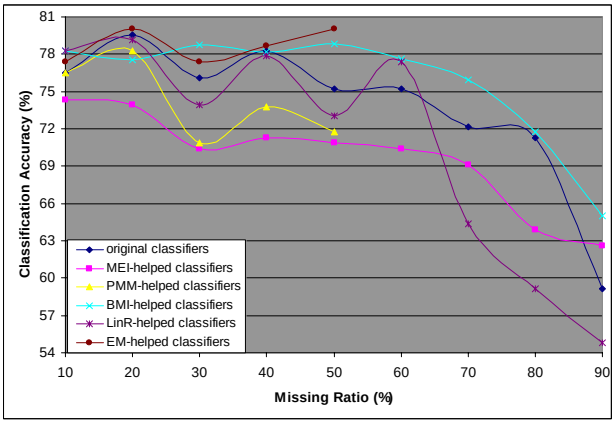

Figure 7. Learning curve of different imputation-helped classifiers with NB on the dataset "australian" with MCAR missing ratios from 10% through 90%

Although second to the EM imputer, the BMI imputer is robust to datasets with missing ratio higher than 50%, for which EM and PMM often fails to produce imputations due to an eigenvalue calculation exception (in Figure 7, we only have the results for datasets with missing ratios at most 50% for EM-helped and PMM-helped classifiers).

BMI is also very efficient, requiring only about five minutes to impute the dataset "letter" (with 20,000 samples for training and test sets and 16 attributes, on our computer that has a 3.2 GHz Intel CPU and 4GB RAM) at the missing ratio of 50%. However, PMM required more than 20 minutes, and EM more than 30 minutes using the same machine.

We did not work on datasets with nominal attributes because BMI, LinR, and PMM require numeric or ordinal values. As NMAR missing data are usually problem specific, we did not experiment on datasets with this missing mechanism. We plan to investigate the impact of using these imputation techniques for MAR incomplete data in our future work.

We see *imputation-helped classifiers with kNN* were extremely helpful for kNN. We believe this is because kNN uses a crude way to handle missing data (see Section 2.2.4), while imputed values can be better used for distance calculation than the missing values.

On average, the original NB classifier can outperform *imputation-helped classifiers with NB* because NB performs classification by computing the class that maximizes a posterior (MAP) based on observed values and the imputed data often does not provide better MAP inference.

For the other classifiers, especially on incomplete data with high missing ratios, we found that classifiers learned on imputed data from a high-quality imputer (e.g., EM-helped classifiers) had better classification performance.

4. Conclusions

The accuracy of classifiers produced by machine learning algorithms generally deteriorates if the training data is incomplete, and preprocessing this data using simple imputation methods, such as mean imputation (MEI), does not generally produce much better classifiers. We therefore proposed and investigate the performance of imputation-helped classifiers, which preprocess incomplete data using accurate imputation techniques (such as Bayesian multiple imputation (BMI) and Expectation Maximization (EM)) to first fill in missing values before giving the completed data to a conventional machine learning algorithm. Empirical results over MCAR missing data show that EM-helped and BMI-helped classifiers generally outperform the original classifiers and other imputation-helped classifiers, including MEI-helped, PMM (predictive mean matching)-helped, and LinR (linear regression)-helped ones. Specifically, EM-helped and BMI-helped classifiers significantly outperform the original classifiers *kNN, neural network*, *one rule*, *decision table*, and *SVM* on incomplete data over all missing ratios we investigated. EM-helped classifiers perform better than other classifiers *logistic regression*, *decision tree (C4.5)*, *random forest* and *PART* on incomplete data with high missing ratios.

Acknowledgement

We thank the members of Data Mining and Machine Learning Lab of Florida Atlantic University. RG is supported by Alberta Ingenuity Centre for Machine Learning and NSERC.

References

[1] Allison, P.D. Multiple Imputation for Missing Data: a Cautionary Tale, *Sociological Methods and Research*, 28: 301-309, 2000.

[2] Blake, C., and Merz, C. UCI repository of machine learning databases, 2000. http://www.ics.uci.edu/~mlearn/MLRepository.html

[3] Breiman, L. Random Forests. Machine Learning 45(1): 5-32, 2001.

[4] Cessie, S. le, and Houwelingen, J.C. van. Ridge Estimators in Logistic Regression, *Applied Statistics*, 41(1), pp. 191-201, 1992.

[5] Dempster, A.P., Laird, N.M., Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society*, B 39, pp. 1-38, 1977.

[6] Frank, E., and Witten, I.H. Generating Accurate Rule Sets Without Global Optimization. ICML 1998.

[7] Holte, R.C. Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11, pp. 63-91, 1993.

[8] Ishibuchi, H., Miyazaki, A., Kwon, K., and Tanaka H. Learning from Incomplete Training Data with Missing Values and Medical Application, IJCNN, pp. 1871-1874, 1993.

[9] John, G.H., and Langley, P. Estimating Continuous Distributions in Bayesian Classifiers. UAI, pp.338-345, 1995.

[10] Kalousis, A., and Hilario, M. Supervised knowledge discovery from incomplete data, ICDM, 269-278, 2000.

[11] Kohavi, R. The Power of Decision Tables. In Proc European Conference on Machine Learning, 1995.

[12] Landerman, L.R., Land, K.C. and Pieper, C.F. An Empirical Evaluationof the Predictive Mean Matching Method for Imputing Missing Values. *Sociological Methods & Research* 26: 3-33, 1997.

[13] Little, R.J.A., and Rubin D.B.: Statistical Analysis with Missing Data. Series in Probability and Mathematical Statistics,Wiley, pp. 278, 1987.

[14] Platt, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization. Advances in Kernel Methods - Support Vector Learning, 1998.

[15] Quinlan, J.R. C4.5: Programs for Machine Learning. Morgan Kaufmann.

[16] Rubin, D.B. Multiple Imputation for Nonresponse in Surveys. *J. Wiley & Sons*, New York, 1987.

[17] Schafer, J.L. Analysis of Incomplete Multivariate Data, *New York: Chapman and Hall*, 1997.

[18] Schneider, T. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. Journal of Climate, 14, pp. 853–871, 2001

[19] Schuurmans, D., and Greiner R. Learning Default Concepts, Canadian Conference on Artificial Intelligence (CAI), 1994.

[20] Witten, I.H. and Frank, E. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[21] Zio, M.D., and Guarnera, U. A Semiparametric Predictive Mean Matching: An Empirical Evaluation, *Conference of European Statisticians*, 2006.