https://doi.org/10.20965/jaciii.2017.p0803

Paper:

When Partly Missing Data Matters in Software Effort Development Prediction

Bhekisipho Twala

Department of Electrical and Electronic Engineering Science, University of Johannesburg
P.O. Box 524, Auckland Park, Johannesburg 2006, South Africa
E-mail: btwala@uj.ac.za

[Received March 2, 2017; accepted June 29, 2017]

The major objective of the paper is to investigate a new probabilistic supervised learning approach that incorporates "missingness" into a decision tree classifier splitting criterion at each particular attribute node in terms of software effort development predictive accuracy. The proposed approach is compared empirically with ten supervised learning methods (classifiers) that have mechanisms for dealing with missing values. 10 industrial datasets are utilized for this task. Overall, missing incorporated in attributes 3 is the top performing strategy, followed by C4.5, missing incorporated in attributes, missing incorporated in attributes 2, missing incorporated in attributes, linear discriminant analysis and so on. Classification and regression trees and C4.5 performed well in data with high correlations among attributes while *k*-nearest neighbour and support vector machines performed well in data with higher complexity (limited number of instances). The worst performing method is repeated incremental pruning to produce error reduction.

Keywords: missing data, software effort prediction, decision tree imputation

1. Introduction

Accurate and unbiased effort prediction is one important contributor to effective software project management. In fact, knowledge of accurate effort estimates in the software project life cycle enables project managers to exploit resources more efficiently. In recent years classifiers have been employed to predict software effort using historical datasets. It is also generally accepted that the highest accuracy results that a supervised learning system (classifier) can achieve depends heavily on the quality of data and the appropriate selection of a learning algorithm for the data. One of the central tasks of classifiers is classifying instances from some domain of application, i.e., determining whether a particular instance belongs to a specified class, given a description of that instance (classification). Another is to use the classification model to predict the response outcome for a new instance (prediction). The wealth and complexity of industrial data lends itself well to the application of classifiers for prediction or classification of software projects according to factors that influence software effort rates [1, 2].

Virtually all research on supervised learning addresses the task of learning to predict or classify complete domain instances. However, in some research situations we often have to classify (or predict) instances given incomplete or noisy vectors. The frequency of incomplete or noisy data (poor data quality phenomena) is one of the most vexing problems for engineering and science researchers, especially those dealing with software data. In fact, the quality of the results obtained from a classifier depends heavily on the quality of the attributes employed to train the classifier. Incomplete software data could be caused by administrative error, defective technique, or technology failure. Another complication could be project managers who flatly refuse to participate in the study. Some researchers follow the practice of flagging readings that are suspect, and these may be converted to missing values or otherwise excluded from the analysis before proceeding.

Recent research has shown that missing values in either the training data or test (unseen) data affect prediction accuracy of learned classifiers [1, 3–18]. The task of learning an accurate incomplete data classifier from instances raises a number of new issues some of which have not been properly addressed, especially by software engineers. There are, however, a few published works or empirical studies in software engineering research that have looked at the incomplete data problem. Some of the studies have used traditional methods [19–23] while others have looked at much more advanced supervised learning methods [1, 24–29]. Other researchers have focussed on classification of incomplete data in other fields like data mining or knowledge discovery.

All the above strategies (especially those employed in empirical software engineering) have failed to make the best use of both the observed and unobserved values when in fact an incomplete instance may already contain enough and, sometimes, valuable information for model construction.

To make headway as to how incomplete data should be handled, the proportion of missing data in key variables becomes critical. This is also directly related to the quality of statistical inference. However, there is no estab-

lished cut-off from the literature regarding an acceptable percentage in a dataset for valid statistical inference [30, 31]. Then, the monotonic and non-monotonic patterns of missing values become equally important. This could be univariate, monotone or arbitrary. Basically, missing data patterns addresses the question as to which values are missing. Finally, the types of processes that can cause an instance to have missing attribute values have to be considered. These include: whether this omission is randomly missing, uninformative, partially informative, or even misleading. Say, for example, one is modelling software development effort per lines of code. There could be no particular reason why a project manager failed to give any information regarding lines of code or a software development effort report, resulting to missing completely as random (MCAR) data or software effort information could be missing due to the number of lines of code; this type of mechanism is called missing at random (MAR). Informatively missing (IM) data could be caused by a project manager deliberately withholding information on projects with high software development effort rates.

The major contribution of the paper is the development and assessment of the robustness of a new decision tree-based probabilistic strategy that incorporates "missingness" in attributes 3 (MIA_3) in terms of predictive accuracy using ten industrial datasets. The performance of MIA_3 is compared with ten state-of-the-art supervised learning (SL) methods and two of the original "missingness" incorporated in attributes strategies (MIA and MIA_2) [8, 13]. Between these SL methods they generate different model forms: linear models, trees, rules and networks. We also note that although some of these methods including C4.5, classification and regression trees (CART) have their own internal approaches of handling unknown attribute values; it is not clear how they would react to external imputation methods. Furthermore, missing values in data could be missing in either the attribute variables or class labels or both. For the purposes of this study, we assume that class labels are non-missing.

The rest of the paper is organised as follows: Section 2 presents details of nine SL methods (which we shall now refer to as classifiers) that are used in this paper. The framework of the suggested MIA_3 is also introduced and described in Section 3. Section 4 empirically evaluates the robustness and accuracy of the nine classifiers found in the literature in comparison with MIA_3 using real-world industrial datasets. We close with a discussion and conclusions, and then highlight directions for future research.

2. Existing Supervised Learning Imputation Methods

2.1. Algorithm Quasi-Optimal

The AQ15 [32] inductive incremental learning system is the descendant of the algorithm quasi (AQ) family. It is based on the logic quasi-optimal solution of the general covering problem and uses the accuracy of the induced rule on the training set. Classification of an unknown instance is performed by matching each rule with that particular instance, and then selecting those that it satisfies. If there is only one such rule, its class is assigned to the instance; otherwise a "default rule" is used (i.e., assigning the instance to the class that occurs most frequently in the entire training set, or among those instances not covered by any rule). Finally, if more than one rule covers the example, one strategy is to order rules into a "decision list", and select only the first rule that fires. When dealing with incomplete data, AQ15 incorporates a bias against missing values into a rule building process; any test of an attribute whose value is unknown (missing) returns a failure, so that the learner focuses on completely known (non-missing) features in selecting rule pre-conditions. The assumption made about the law generating the missing values when using AQ15 is that the data is MCAR.

2.2. Artificial Neural Network

ANNs have been used for a variety of classification and regression (prediction) problems including pattern and speech recognition [33], credit risk prediction [34], and so on. There are many types of ANNs, but for the purposes of this paper we shall concentrate on the multi-layer perceptrons also known as backpropagation neural networks. Backpropagation neural networks perform a hill-climbing search procedure on the weight space described above or a (noisy or stochastic) gradient descent numerical method whereby an error function is minimized. At each iteration, each weight is adjusted proportionally to its effect on the error. One cycle through the training set and on each example changes each weight proportionally to its effect on lowering the error. One may compute the error gradient using the chain rule and the information propagates backwards through the network through the interconnections, which accounts for the procedure's name. The final product of this activity is a trained neural network (also known as the ultimate "black box"). ANNs require complete instances for analysis. Hence, the datasets containing missing values are processed using the ANN (data preprocessing) prior to them being supplied to the learning algorithm. This is based on the assumption of MCAR.

2.3. C4.5

C4.5 is an algorithm used to generate a decision tree by Quinlan [35]. Decision trees (DTs) are non-parametric used for regression and classification. C4.5 uses the concept of information entropy and the pruning rule to create a model that predicts the value of a target variable by learning simple decision rules inferred from data vectors. Quinlan's probabilistic approach of dealing with incomplete data involves "fractioning" of cases based on a priori probability of each value determined from the instances at that node that have specified values. First, the information gain measure is penalised by the proportion of unknown instances and then these instances are split to both subnodes. For classification, Quinlan's approach

is to explore all branches below the node in question and then take into account that some branches are more probable than others. The weights of the instance fragments classified in different ways at the leaf nodes of the tree are summed and then the class with the highest probability or the most probable classification is chosen. Since C4.5 does not consider dependencies among the attributes, we shall thus assume a MCAR mechanism.

2.4. Classification and Regression Trees (CART)

One other sophisticated but yet refined DT-based supervised learning imputation method worthy of note and study is the surrogate variable splitting (SVS), which has been used for the CART system [36]. The whole idea of SVS is to use surrogate variables to impute the predictor values that are missing. CART handles missing values in the database by substituting "surrogate splitters." Surrogate splitters are predictor variables that are not as good at splitting a group as the primary splitter but which yield similar splitting results in terms of association or correlation (i.e. how close the variables are related); they mimic the splits produced by the primary splitter with the highest association; the second best, the third best, and so on. The surrogates are used for tree nodes whenever there are missing values. When dealing with incomplete data, the CART system relies heavily on the correlation or associations of the attributes, hence, is driven by the MAR mechanism.

2.5. *k*-Nearest Neighbor (*k*-NN)

k-NN is one of the most venerable algorithms in statistical estimation and pattern recognition. The way in which the algorithm decides which of the points from the training set are similar (in terms of distance functions) enough to be considered when choosing the class to predict for a new observation is to pick the k closest data points to the new observation, and to take the most common class among these [37–39]. The prediction can be the mode, average, or some interpolation between the prediction of these k training instances, perhaps weighting closer instances more than distant instances. For this method to work, distance metrics such as Euclidean, Manhattan or Minkowski (for continuous attributes) or Hamming distance (for categorical attributes) are required that measures the closeness of two instances. Of late, such an algorithm has become popular in imputing missing data whereby instances with similar characteristics to the instance of interest are used to impute missing values [5, 12, 40]. *k*-NN requires that data are MCAR.

2.6. Linear Discriminant Analysis (LDA)

LDA is a classification method that finds a linear transformation ("discriminant function") of two predictors, that yields a new set of transformed values that provides a more accurate discrimination than either predictor alone [41] LDA tries to optimize class separability and keeping the variance of all classes roughly constant by reducing the dimensionality of the data while preserving as much of the class discriminatory information as possible. LDA looks for a projection where instances from the same class are projected very close to each other and, at the same time the projected means are as further apart as possible. The Mahalanobis distance between two groups has been one way of assessing the effectiveness of the discrimination. When dealing with incomplete data, LDA uses a mean imputation strategy, i.e. replacing missing values of an attribute with the mean of the attribute. This strategy is applicable for continuous data. For discrete data of the corresponding attribute, the most frequent value was utilised. LDA is based on the assumption that data is MCAR.

2.7. Naïve Bayes Classier (NBC)

The NBC (which is based on Bayes' theorem) learns from the training data, the conditional probability of each attribute (predictor) given the class (target) label [42–46]. The major strong assumption of NBC is that all attributes are independent given the value of the class (class conditional independence). Classification is therefore done applying Bayes rule to compute the probability of C given $A_1, \ldots, A_n$ and then predicting the class with the highest posterior probability. The posterior probability can be calculated by first, constructing a frequency table for each attribute against the target, then transforming the frequency tables to likelihood tables, and finally calculate the posterior probability for each class. The assumption of conditional independence of a collection of random variables is very important for the above result. Otherwise, it would be impossible to estimate all the parameters without such an assumption. To perform imputation, we treat each attribute that contains missing values as the class attribute, then fill each missing values for the selected class attribute with the class predicted from the conditional probabilities established during training. The NBC assumes that data is MCAR due to the independence of the attributes given the class target.

2.8. Repeated Incremental Pruning to Produce Error Reduction (RIPPER)

RIPPER by Cohen [47] is a fast and effective rule-based learning algorithm that builds a set of rules that identify classes while minimizing the amount of error. The rule-growing algorithm begins with an empty condition, and greedily adds conditions until the rule no longer makes incorrect prediction on the growing set. Here, each condition represents the amount of software development effort for a specific project. Next, the learned rule is simplified by deleting conditions so as to improve performance of the rule on the pruning set. All samples covered by the formed rule are then removed from the training set and a new rule is learned in the same way until all examples are covered by the rule set. The error is defined by the number of instances misclassified by the rules. RIPPER incorporates a bias against missing values into a rule building process; any test of an attribute whose value is

unknown (missing) returns a failure, so that the learner focuses on completely known (non-missing) features in selecting rule pre-conditions. The assumption made about the law generating the missing values when using RIPPER is that the data is MCAR.

2.9. Support Vector Machine (SVM)

The principal goal of the SVM approach is to fix the computational problem of predicting with kernels [48]. The main idea is to determine a classifier or regression machine which minimizes the empirical risk (i.e., the training set error) and the confidence interval (which corresponds to the generalisation or test set error). In other words, the idea is to fix the empirical risk associated with architecture and then use a method to minimize the generalisation error. SVM performs classification by finding the hyperplane that maximizes the margin between, say, two classes. The vectors (instances) that define the hyperplane are the support vectors. Motivated by statistical learning theory, SVMs have successfully been applied to numerical tasks, including regression and classification. They can perform both binary classification (pattern recognition) and real valued function approximation (regression estimation) tasks. The standard formulation of SVMs does not allow for classification with incomplete data. Hence, for the handling of missing values in SVM classifiers, the maximal variation approach by Pelckmans [49] is followed in this article.

3. New Supervised Learning Imputation Method

MIA_3 introduces a number of extensions (in terms of the number of splits at each internal node of the decision tree) from the original MIA [50] and the later revised MIA_2 [51]. The key differences between the three approaches are explained below.

The MIA_3approach is very simple, natural and closely related to the technique of treating "missing" as a category in its own right, generalizing it for use with continuous as well as categorical variables. It is applicable to any method of constructing DTs, regardless of that method's detailed splitting/stopping/pruning rules. The two approaches are the same for categorical attributes, but differ a little in their treatment of continuous attributes: rather than categorizing continuous variables, we incorporate missingness directly in splits of continuous variables. This approach can also be expected to be particularly useful when missingness is not random but informative. Classifying a new individual whose value of a branching attribute is missing is immediate provided there was missingness in that attribute in the training set that led to the decision tree [52]. It utilizes a DT as described below.

An unknown (missing) value is considered an additional attribute value. Hence, the number of values is increased by one for each attribute that depicts an unknown value in the training or test set.

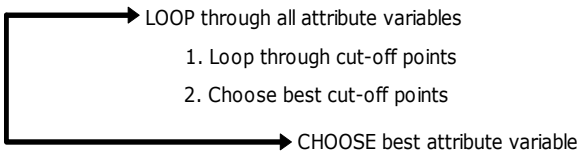

Fig. 1. Standard algorithm for feature selection.

If X_n is an ordered or numeric attribute variable with unknown attribute values, the proposed approach searches essentially over all possible values of x_n for binary splits of the following form:

1. Split A: (X_n = observed) versus (X_n = missing)
2. Split B: ($X_n \leq x_n$) versus ($X_n > x_n$ or X_n = observed)
3. Split C: ($X_n \leq x_n$ or X_n = observed) versus ($X_n > x_n$)
4. Split 4: (X_n = missing) versus (X_n = observed)
5. Split E: ($X_n \leq x_n$) versus ($X_n > x_n$ or X_n = missing)
6. Split F: ($X_n \leq x_n$ or X_n = missing) versus ($X_n > x_n$)

The idea is to find the best split from the candidate set of splits given above, with the goodness of split measured by how much it decreases the impurity of the sub-samples.

If X_n is a nominal attribute variable (i.e., a variable that takes values in an unordered set), the search is over all splits of the form:

1. Split A: (X_n = observed) versus (X_n = missing) where Y_n is the splitting subset at node n.
2. Split B: ($X_n \in \underline{Y}_n$) versus ($X_n \notin Y_n$ or X_n = observed)
3. Split C: ($X_n \in \underline{Y}_n$ or X_n = observed) versus ($X_n \notin Y_n$)
4. Split D: (X_n = missing) versus (X_n = observed)
5. Split E: ($X_n \in \underline{Y}_n$) versus ($X_n \notin Y_n$ or X_n = missing)
6. Split F: ($X_n \in \underline{Y}_n$ or X_n = missing) versus ($X_n \notin Y_n$)

When the training set did not have any missing values for some attributes, the above reduces to using a standard DT split for such variables, i.e., $X_n \leq x_n$ versus $X_n > x_n$ (for an ordered attribute variable) or $X_n \in \underline{Y}_n$ versus $X_n \notin Y_n$ (for a categorical attribute variable). So, if there were ∂ options for splitting a branch without missingness, there are $2\partial + 1$ options to be explored with missingness present.

The standard algorithm for feature selection and the proposed algorithm for feature selection with unknown (missing) attribute values when using DTs are displayed in **Figs. 1** and **2**, respectively. The algorithm works in the same way to determine the outcome of the test when classifying a new instance, and given that at that particular internal node there are attribute values missing as it was the case with learning. If the unseen instance is regular

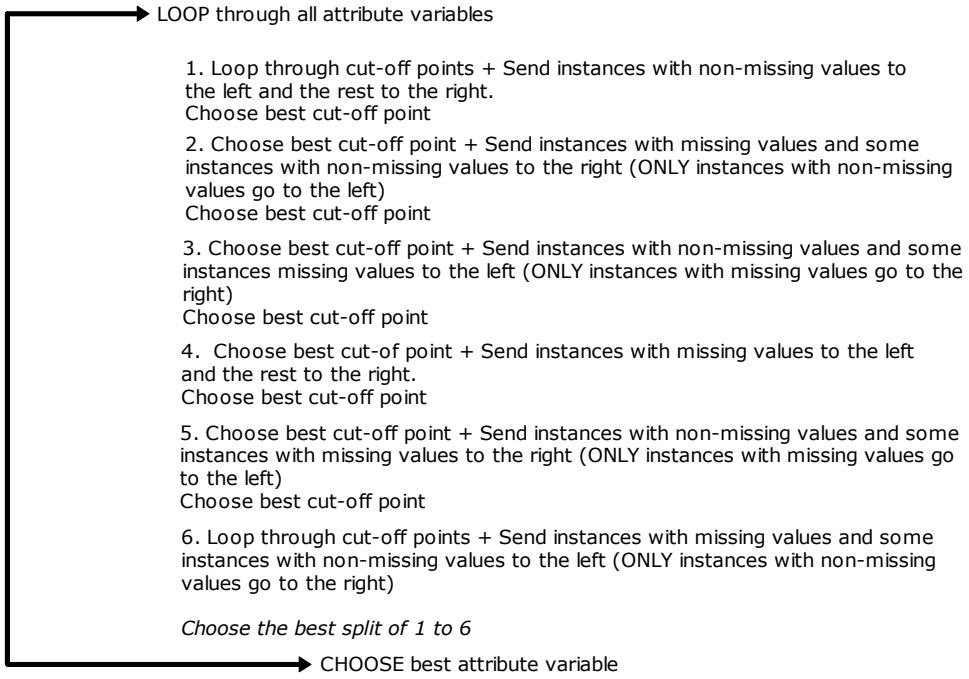

Fig. 2. New algorithm for feature selection with missing attribute values.

(without any unknown attribute value) then the classification is carried out the traditional way. However, if an instance involves one or more unknown values, then the algorithm tries in turn all the six binary splits and selects the best split. The split chosen determines the number of instances branching on each path at a particular node for which a value is missing.

The key differences between MIA, MIA_2 and MIA_3 is in relation to the number of splits as summarized in **Fig. 2**; MIA only is based on three splits (1, 2, and 3) while MIA2 is based on only four splits (1, 2, 3, and 4). For more details about MIA and MIA_2, the reader is referred to Twala [7, 13, 50].

4. Experiments

4.1. Experimental Setup

In order to empirically evaluate the performance of the new technique, an experiment is used on ten industrial datasets [7, 51, 53, 54] in terms of the smoothed error rate (SER). The SER is used due to its variance reduction benefit. Instead of summing terms that are either zero or one as in the error-count estimator, the smoothed estimator uses a continuum of values between zero and one in the terms that are summed. The resulting estimator has a smaller variance than the error-count estimate (which is also known to be optimistically biased for most classifiers).

MIA_3 is empirically evaluated with eleven base methods of classifier construction with incomplete data (ANN, AQ15, C4.5, CART, k-NN, LDA, NBC, RIPPER, SVM, MIA and MIA_2) using the 10 industrial datasets and in terms of the smoothed error rate. These classifiers were chosen for a number of reasons. First, each utilizes a different from of parameter estimation/learning. Second, between them they generate three different model forms: linear models, trees and networks. Third, they are practically applicable within the software engineering industry. For all the existing classifiers (with the exception of MIA, MIA_2 and MIA_3), the Waikato environment for knowledge analysis (WEKA) software (with default parameters) was utilised as a tool of choice to perform classification [55]. A MATLAB [56] code was developed for MIA_3.

A brief definition of the SER (based on the available posterior probability error rate estimates) is given below.

We have a set of n instances in the training sample, $S = \{x_i, y_i | i = 1, \ldots, n\}$, where x_j is a vector containing d attributes, and $y \in \{1, 2, \ldots, m\}$ is a class label given m classes. Let n_t be the number of instances in S in class t. Further let R_t be a set of instances such that the posterior probability belonging to class t is the largest, and R_{ut} be the set of instances from class u such that that the posterior belonging to class t is the largest. The classification error rate for class t is defined as [57]:

$$e_t = \int_{R_t} f_t(x)\, dx$$

The posterior probability of an instance x for class t can be written as

$$p(t|\mathrm{x}) = \frac{q_t f_t(\mathrm{x})}{f(\mathrm{x})}$$

where $f(x) = \sum_u q_u f_u(x)$ is the estimated unconditional density of x; q_t is the prior probability of as instance x belonging to class t.

Thus, if you replace $f_t(x)$ with $p(t|x) f(x)/q_t$, the

Table 1. Datasets used for the software engineering problem.

Dataset	Instances	Attributes		Class Distribution
		Numerical	Categorical	
Kemerer	18	4	2	1 (24.1%); 2 (75.9%)
Bank	18	2	7	1 (84.3%); 2 (15.7%)
Test equipment	16	17	4	1 (43.0%); 2 (57.0%)
DSI	26	5	0	1 (49.9%); 2 (50.1%)
Moser	32	1	1	1 (42.2%); 2 (57.8%)
Desharnais	76	3	6	1 (15.7%); 2(84.3%)
Experience	95	1	5	1 (57.0%); 2 (43.0%)
IBSG-version 7	166	2	7	1 (35.6%); 2 (64.4%)
CCCS	282	8	0	1 (48.2%); 2 (51.8%)
Company X	10434	4	18	1 (33.0%); 2 (29.2%); 3 (37.8%)

smoothed classification error rate is:

$$e_t = 1 - \frac{1}{q_t} \int_{R_t} p(t|x) f(x) dx$$

Each dataset, defines a different learning problem as summarized in **Table 1**.

To maintain a level of consistency with the proportion of missing values simulated, only datasets whose percentage "missings" came out to be close to the nominal percentage missing were simulated. Otherwise, those that were not close were rejected and not considered in the analysis. To carry out this task, some form of truncated binomial distribution was used, i.e., any percentage value that was outside the original binomial and truncated binomial distribution regions was rejected. Any value less than or greater than 0.5% to the specific level of missingness being looked at was not considered in our analysis.

Three suites of data were created corresponding to MCAR, MAR and IM. For MCAR, the random generator is used while for MAR and IM, a quintile attribute-pair approach is utilized. For MAR, the idea is to condition the generation of missing values based upon the distribution of the observed values. Attributes of a dataset are separated into pairs, say, (A_X, A_Y), where A_Y is the attribute into which missing values are introduced and A_X is the attribute on the distribution of which missingness of A_Y is conditioned. For example, to generate missingness in half of the attributes for a dataset with, say, 12 attributes ($A_1,\ldots,A_{12}$), the pairs (A_1,A_2), (A_3,A_4) and (A_5,A_6) could be utilized. We assume that A_1 is highly correlated with A_2; A_3 highly correlated with A_4, and so on. For the (A_1,A_2) pairing, A_1 is used to generate a missing value template of zeros and ones utilizing the quintile approach. The template is then used to "knock off" values (i.e., generating missingness) in A_2, and vice versa.

For conditions with IM data, a procedure identical to MAR was implemented. However, for IM, the missing values template was created using the same attribute variable for which values are deleted in different proportions. Both of these procedures have the same percentage of missing values as their parameters. These two approaches were also run to get datasets with four levels of proportion of missingness p. The experiment consists of having p% of data missing from only the testing (classification) set.

For each dataset, two missing data patterns (suites) were created. First, missing values were simulated on half of the attributes (MCAR_{half}, MAR_{half}, and IM_{half}). Second, missing values were introduced on all the attribute variables (MCAR_{all}, MAR_{all}, and IM_{all}). For both suites, the missingness was evenly distributed across all the attributes. To measure the performance of methods, the training set-test set methodology is employed whereby the missing values are simulated in both sets. For each run, each dataset is split randomly into 80% training and 20% testing, with different percentages of missing data (i.e., 10%, 15%, 25%, 40%, 55%) in the covariates for testing set. 5-fold cross validation was used for the experiment.

It was also reasoned that the condition with no missing data should be used as a baseline and what should be analyzed is not the error rate itself but the increase or excess error induced by the combination of conditions under consideration. Therefore, the excess error is the error achieved by a SL method given that the dataset is incomplete less the error exhibited by the SL method given that dataset is complete.

Analyses of variance, using the general linear model procedure, were used to examine the significance of the main effects and their respective interactions. The comparison of means was conducted by using the Tukey post hoc test, and utilizing the MINITAB software [58, 59]. This was done using a 5-way factorial design experiment, with four fixed effect factors: twelve classifiers; two types of missing data patterns (i.e. number of attributes with missing values); five levels of missing data proportions; three types of missing data mechanisms; and five replicates. The 10 datasets is the only random effect factor. This makes it a total of 18,000 experiments (i.e. $12 \times 2 \times 5 \times 3 \times 5 \times 10$). The results were then averaged across 5 folds of the cross-validation process (replicates) before carrying out the statistical analysis. The averaging was done as a reduction in error variance benefit

4.2. Experimental Results

Main Effect

Table 2. Overall means.

Fixed effects	Smoothed error rate
Supervised learning methods:	
1. ANN	40.6
2. AQ15	37.4
3. C4.5	27.5
4. CART	37.4
5. k-NN	36.9
6. LDA	33.2
7. MIA	29.3
8. MIA2	28.1
9. MIA3	**22.4**
10. NBC	41.3
11. RIPPER	47.2
12. SVM	45.1
Missing data proportions:	
1. 0%	**22.6**
2. 10%	23.1
3. 15%	23.5
4. 25%	26.7
5. 40%	31.1
6. 55%	35.9
Missing data patterns:	
1. Half	**24.6**
2. All	29.3
Missing data mechanisms:	
1. MCAR	**19.3**
2. MAR	24.5
3. IM	33.4
Random effects	
Ten industrial datasets: (See **Table 1**)	

The mixed-effects of analysis of variance reveals a significant difference (p-value $= 0.0082$) in performance among supervised learning methods at the 5 % level. The pairwise comparison test shows that MIA_3 (one the one hand) is significantly higher (p-value $= 0.0091$) compared to the other eleven methods (on the other hand). This is the case at the 5% level of significance.

Increases in the proportion of missing values in both the training and test sets being associated with increases in error rates are presented in **Table 2**.

Furthermore, it appears that for all the methods missing values have a greater effect on software effort predictive accuracy when they are distributed among all the attributes compared with when the missing values are in half of the attributes. The difference in error rate between the two conditions is about 5%. Lastly, the results show all the methods performing worse under the IM condition compared with when data are MCAR or MAR. However, all the methods appear to deal with MCAR data more effectively than MAR data with an error rate increase difference of about 5.2% compared with a much bigger difference of about 14.1% for the MCAR and IM conditions in error rates. In fact, the error rate increase when 50% of values are missing in both the training and test sets is about one and a half times as big as the error rate increase when 10% of values are missing on both sets.

Interaction effect

Figure 3 summarises the overall excess error rates for current and new methods against the proportion of missing data, the pattern of missing data and the mechanisms generating the missing values. The smoothed error rates of each supervised learning method are averaged over 10 datasets. All error rates are increases over complete data case formed by taking differences. From these experiments the following results are observed.

From **Fig. 3** it is clear that when both the training and testing data is incomplete due to the $MCAR_{half}$ mechanism, MIA_3 performs better than the other methods. This is the case at all levels of missing values. The overall worst performance is by RIPPER, which is closely followed by k-NN and AQ15, respectively. The differences in performance are mostly significant at the 5% level of significance, especially with increases in the proportion of missing values. For $MCAR_{all}$ data, the performance of all the methods follows a similar pattern to the results observed for the $MCAR_{half}$ suite. However, the difference in performances for the former is now much more noticeable, with C4.5 outperforming MIA_3 (at the 40% and 55% levels of missing values.

The results show MIA_3 as a more effective method for handling MAR_{half} data, especially at lower levels of missing values and at the 55% level (**Fig. 3**). For this kind of missing mechanism suite, the performance of LDA and C4.5 improves as the amount of missing values increases. Experimental results for the twelve methods' handling of MAR_{all} data, show MIA_3 achieving higher accuracy at three levels of missing values (i.e. 10%, 25% and 55%). However, the performance of all the methods degrades when data is MAR_{all}, with CART, RIPPER and SVM exhibiting high error rates. Good performance is observed for AQ15 when dealing with MAR_{all} data.

The results in **Fig. 3** show IM_{half} data as more damaging to SVM than any other method (with the exception of RIPPER). With 55% of missing values, SVMs error rate increases from 18.2% (MAR_{half}) to 28.4% (IM_{half}). The best performance is once again by MIA_3. At lower levels of missing values, there appears to be no significant difference in performance between most of the methods. However, as the amount of missing values increases, the difference in performances becomes significant at the 5% level.

It can also be seen from **Fig. 3** that results yielded by methods for IM_{all} data are identical to results achieved by methods for MAR_{all} data. The only difference is the performance of CART which drops from being the third best method (for dealing with IM_{half} data) to being one of the worst methods (for dealing with IM_{all} data). This is the case at the 10% level of missing data. AQ15 appears to deal with IM_{all} data better than IM_{half} data, especially at higher levels of missing data. The best performance is by MIA_3.

It seems that the overall performance of MIA_3 is rather effective on average compared with the other strategies including the ones that incorporates "missingness" in attributes (MIA and MIA_2). This is the case for all the three missing data mechanisms. In addition, these three strategies deal with IM data more effectively than any other

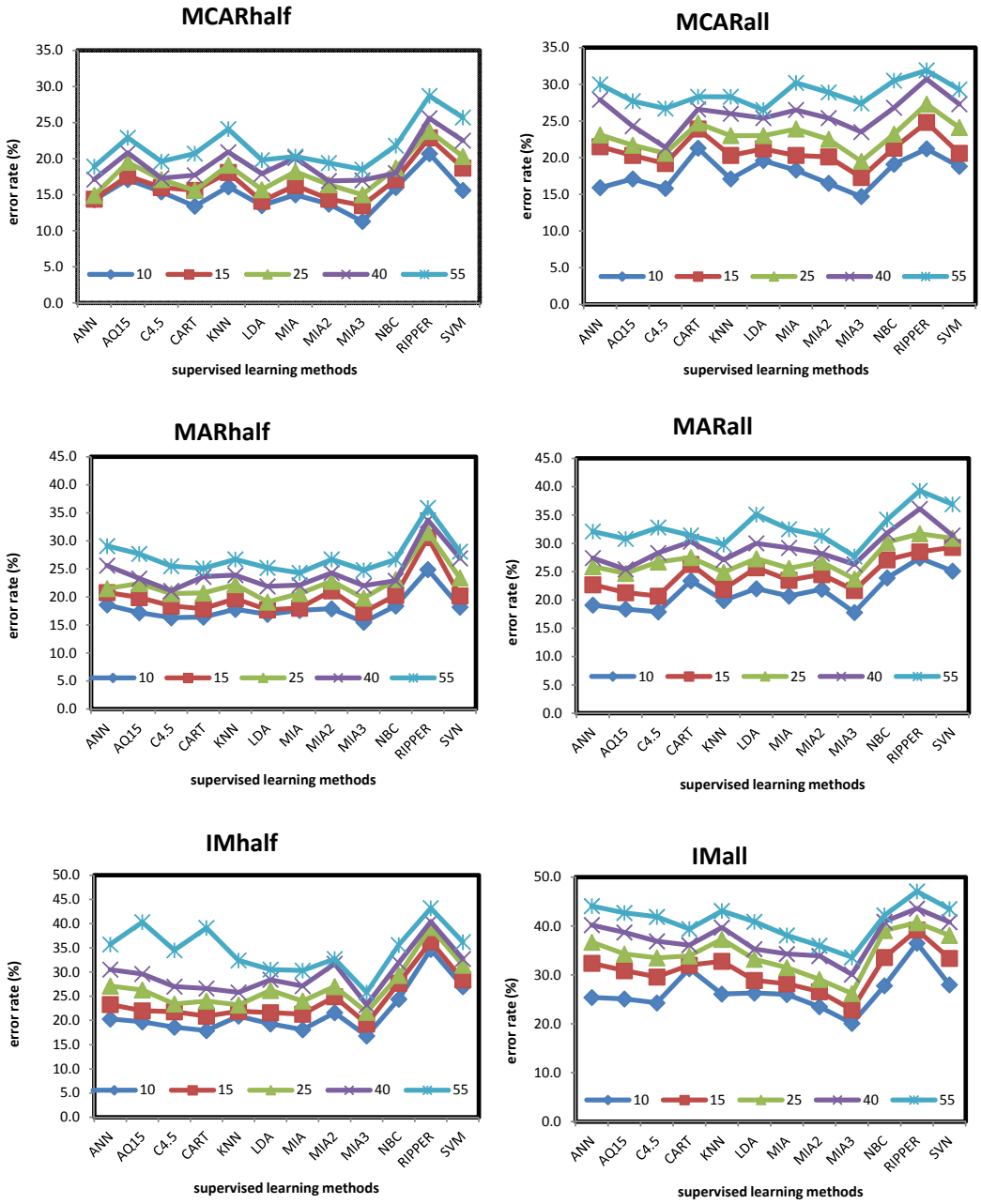

Fig. 3. Impact of missing data proportions, patterns and mechanisms on predictive accuracy.

method in all our experiments. The slightly better performance of AQ15 compared with RIPPER is rather surprising considering that they are both rule-based strategies.

Selected individual datasets

Figure 4 summarizes the error rates exhibited by three of the best supervised learning methods from our previous experimental results.

This is only for a selected number of datasets described in terms of their respective characteristics (i.e., total number of instances, number of categorical or numerical attributes, number of classes and the class distribution percentage). All three methods appear to deal with the Company X problem better, with MIA_3 exhibiting the lowest error rate across the five datasets. This is the only datasets with more than 10, 000 instances in our experiments. The performances of all the methods appear to degrade with decreases in the number of instances for each dataset. The differences in performances are significant at the 5% level. LDA appears to only outperform C4.5 for the CCCS and company X data problems.

5. Conclusion

Data quality has become a very important component of supervised learning. Using a simulation-based study based on software effort data we assessed twelve super-

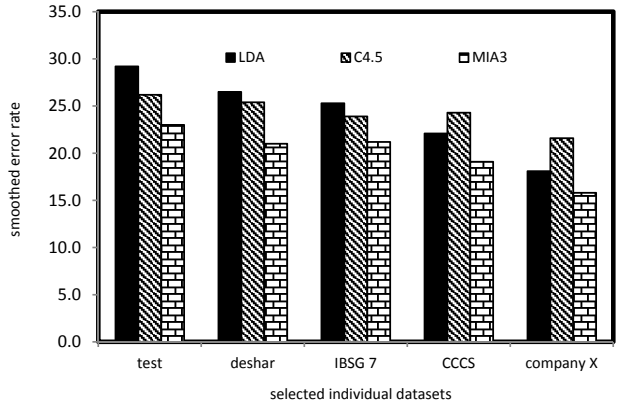

Fig. 4. Overall means for three methods (individual datasets).

vised learning methods for handling up to 50% missing data proportions, patterns, and mechanisms in the context of predictive accuracy. The findings from this simulation study demonstrate that for a software engineering study assessing the association between proportions, patterns and mechanisms of missing data is an advantage especially when accommodating "missingness" (making the best use of both the observed and unobserved values).

The empirical results demonstrated that the proposed method is better than most supervised imputation methods in terms of software effort prediction accuracy and classification error rate. Its performance was more exceptional when dealing with informatively missing data (one of the most difficult mechanisms to deal with in data). We suggest that researchers do not use rule-based approaches when dealing with missing software effort data. Also, caution should be taken when using SVM which achieved very low accuracy rates. Our findings further demonstrate why taking into account both the observed and non-observed values when dealing with the incomplete-data problems becomes important especially for small datasets (which are commonplace in software engineering research).

In practice it may be difficult to estimate the effects of missing data and to identify and separate its sources. Therefore one should take measures against multiple possible missing data effects. We emphasize the importance of further research to better understand the proportions, patterns and mechanisms of missing data in software engineering and to provide statistical guidance to researchers in the field. We further propose that the proposed strategy be tested extensively on on-industrial datasets.

Acknowledgements

The authors would like to thank the Institute for Intelligent Systems at the University of Johannesburg in South Africa for supporting the project.

References:

[1] B. Twala, "Dancing with dirty road traffic accidents data: The case of Gauteng province in South Africa," J. of Transportation Safety and Security, Vol.4, No.4, pp. 323-335, 2014.

[2] P. Winston, "Artificial Intelligence," Addison-Wesley, 3rd ed. Part II: Learning and Regularity Recognition, 1992.

[3] G. H. John, "Robust decision trees: Removing outliers from databases," Proc. of the 1st Int. Conf. on Knowledge Discovery and Data Mining, pp. 174-179, 1995.

[4] A. Kalousis and M. Hilario, "Supervised knowledge discovery from incomplete data," Proc. of the 2nd Int. Conf. on Data Mining 2000, WIT Press, 2000.

[5] G. Batista and M. C. Monard, "An Analysis of Four Missing Data Treatment Methods for Supervised Learning," Applied Artificial Intelligence, Vol.17, pp. 519-533, 2003.

[6] E. Acuna and C. Rodriguez, "The treatment of missing values and its effect in the classifier accuracy," Classification, Clustering and Data Mining Applications, Studies in Classification, Data Analysis and Knowledge Organisation, pp. 639-647, 2004.

[7] B. Twala, "Effective Techniques for Handling Incomplete Data Using Decision Trees," Unpublished Ph.D. thesis, Open University, Milton Keynes, UK, 2005.

[8] B. Twala, M. C. Jones, and D. J. Hand, "Good methods for coping with missing data in decision trees," Pattern Recognition Letters, Vol.29, pp. 950-956, 2008.

[9] B. Twala and M. Phorah, "Predicting Incomplete Gene Microarray Data with the Use of Supervised Learning Algorithms," Pattern Recognition Letters, Vol.31, No.13, pp. 2061-2069, 2010.

[10] B. Twala, "Impact of Noise on Credit Risk Prediction Does Data Quality Matter?," Intelligent Data Analysis, Vol.17, No.6, pp. 1115-1134, 2013.

[11] K. C. Leung and C. H. LEeung, "Dynamic discriminant functions with missing feature values," Pattern Recognition Letters, Vol.34, No.13, pp. 1548-1556, 2013.

[12] S. Huang and Q. Zhu, "A pseudo-nearest-neighbour approach for missing data recovery on Gaussian random sets," Pattern Recognition Letters, Vol.23, No.13, pp. 1613-1622, 2013.

[13] B. Twala, "Reasoning with noisy software effort data," Applied Artificial Intelligence, Vol.28, No.6, pp. 533-554, 2014.

[14] K. Shimada and T. Hanioka, "An Evolutionary Method for Associative Contrast Rule Mining from Incomplete Database," J. Adv. Comput. Intell. Intell. Inform. (JACIII), Vol.19, No.6, pp. 766-777, 2016.

[15] Y. Endo, T. Suzuki, N. Konoshita, and Y. Hamasuna, "On Fuzzy Non-Metric for Data with Tolerance and its Application to Incomplete Data Clustering," J. Adv. Comput. Intell. Intell. Inform. (JACIII), Vol.20, No.4, pp. 571-579, 2016.

[16] K. Lakshminarayan, S. A. Harp, and T. Samad, "Imputation of Missing Data in Industrial Databases," Applied Intelligence, Vol.11, pp. 259-275, 1999.

[17] B. Twala, "Combining Classifiers for Credit Risk Prediction," Journal of Systems Science and Systems Engineering, Vol.18, No.3, pp. 292-311, 2009.

[18] X. Zhu and W. Wu, "Class noise vs. attribute noise: A quantitative study of their impacts," Artificial Intelligence Review, Vol.22, No.3-4, pp. 177-210, 2004.

[19] K. Strike, K. El Emama, and N. Madhavji, "Software cost estimation with incomplete data," IEEE Trans. on Software Engineering, Vol.27, No.1, pp. 890-908, 2001.

[20] I. Myrtveit, E. Stensrud, and U. Olsson, "Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputation Methods and Likelihood-Based Methods," IEEE Trans. on Software Engineering, Vol.27, No.11, pp. 1999-1013, 2001.

[21] D. R. Cox, "Some procedures associated with the logistic qualitative response curve," Research papers in Statistics: Festschrift for J. Neyman (ed. F.N. David), Wiley, pp. 55-71, 1966.

[22] N. E. Day and D. F. Kerridge, "A general maximum likelihood discriminant," Biometrics, Vol.23, pp. 313-323, 1967.

[23] D. W. Hosmer and S. Lameshow, "Applied Logistic Regression," Wiley, 1989.

[24] M. Cartwright, M. Shepperd, and Q. Song, "Dealing with missing software project data," Proc. of the 9th Int. Software Metrics Symp. (METRICS '03), pp. 154-165, 2003.

[25] P. Jönsson and C. Wohlin, "An evaluation of *k*-nearest neighbour imputation using likert data," 10th Int. Software Metrics Symp. (METRICS '04), pp. 108-118, 2004.

[26] Q. Song, M. Shepperd, and M. Cartwright, "A short note on safest default missingness mechanism assumptions," Empirical Software Engineering, Vol.10, pp. 235-243, 2005.

[27] P. Sentas and L. Angelis, "Categorical missing data imputation for software cost estimation by multinomial logistic regression," J. of Systems and Software, Vol.79, No.3, pp. 404-414, 2006.

[28] B. Twala, "Ensemble missing data techniques for software effort prediction," Intelligent Data Analysis, Vol.14, pp. 299-331, 2010.

[29] J. Van Hulse and T. M. Khoshgotaar, "Incomplete-case nearest neighbour imputation in software measurement," Information Science, Vol.259, pp. 596-610, 2014.
[30] R. J. A. Little and D. B. Rubin, "Statistical Analysis with missing data," Wiley, 1987.
[31] J. L. Schafer, "Analysis of Incomplete Multivariate Data," Chapman and Hall, 1997.
[32] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The multipurpose incremental learning system AQ15 and its testing application to three medical domains," Proc. of the 5th National Conf. on Artificial Intelligence, pp. 1041-1045, AAAI Press, 1986.
[33] B. D. Ripley, "Pattern Recognition and Neural Networks," Cambridge University Press, John Wiley, 1992.
[34] D. West, "Neural Network Credit Scoring Models," Computers & Operations Research, Vol.27, pp. 1131-1152, 2000.
[35] J. R. Quinlan, "C.4.5: Programs for machine learning," Morgan Kauffman Publishers, INC, 1993.
[36] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and Regression Trees," Wadsworth, 1984.
[37] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," Machine Learning, Vol.24, pp. 173-202, 1991.
[38] D. J. Hand and V. Vinciotti, "Choosing k for Two-Class Nearest Neighbour Classifiers with Unbalanced Classes," Pattern Recognition Letters, Vol.24, pp. 1555-1562, 2003.
[39] C. C. Holmes and N. M. Adams, "A Probabilistic Nearest Neighbour Method for Statistical Pattern Recognition," J. of the Royal Statistical Society, Series B, Vol.64, pp. 295-306, 2002.
[40] J. Branke, S. Meisel, and C. Schmidt, "Simulated annealing in the presence of noise," J. of Heuristics, Vol.14, No.6, pp. 627-654, 2008.
[41] P. McCullagh and J. A. Nelder, "Generalised Linear Models," 2nd Edition, Chapman and Hall, 1990.
[42] R. Duda and P. Hart, "Pattern Classification and Scene Analysis," John Wiley, 1973.
[43] D. J. Hand, "Construction and Assessment of Classification Rules," Wiley, 1997.
[44] P. Domingos and M. Pazzani, "Beyond independence: conditions for the optimality of the simple Bayesian classifier," Proc. of the 13th Int. Conf. on Machine Learning, pp. 105-112, 1996.
[45] I. Kononenko, "Semi-naïve Bayesian classifier," Proc. of European Conf. on Artificial Intelligence, pp. 206-219, 1991.
[46] P. Langley and S. Sage, "Induction of selective Bayesian classifiers," Proc. Conf. on Uncertainty in AI, Morgan Kauffmann, 1994.
[47] W. W. Cohen, "Fast effective rule induction," Proc. of the 12th Int. Conf. in Machine Learning, Lake Tahoe, California, Morgan Kauffman, 1995.
[48] V. N. Vapkin, "The Nature of Statistical Learning Theory," Springer, 1995.
[49] K. Pelckmans, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Handling Missing Values in Support Vector Machine Classifiers," Neural Networks, Vol.18, pp. 684-692, 2005.
[50] B. Twala, C. Jones, and D. J. Hand, "Good Methods for Coping with Missing Data in Decision Trees," Pattern Recognition Letters, Vol.29, pp. 950-956, 2008.
[51] B. Twala, "Extracting Grey Relational Systems from Incomplete Road Traffic Accidents Data: The Case of the Gauteng Province in South Africa," J. of Expert Systems – The J. of Knowledge Engineering, Vol.31, No.3, pp. 220-231, 2014.
[52] T. Tran, D. Phung, and S. Venkatesh, "Tree-based iterated local search for Markov random fields with application in image analysis," J. of Heuristics, Vol.21, No.1, pp. 25-45, 2015.
[53] C. L. Blake and C. J. Mertz, "UCI Repository of Machine Learning Databases," University of California, Department of Information and Computer Science, Irvine, http://www.ics.uci.edu/˜mlearn/MLRepository.html [accessed Aug. 4, 2014], 1998.
[54] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, "The PROMISE repository of empirical software engineering data," http://promisedata.googlecode.com, West Virginia University, Department of Computer Science [accessed Aug. 4, 2014], 2012.
[55] I. H. Witten and E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques," 2nd Edition, Morgan Kauffmann, Francisco, 2005.
[56] MATLAB, The MathWorks Inc., Natick, MA, 2000.
[57] K. Fukunaga and D. L. Kessel, "Nonparametric Bayes Error Estimation Using Unclassified Samples," IEEE Trans. on Information Theory, Vol.19, pp. 434-440, 1973.
[58] MINITAB, "Statistical Software for Windows 9.0," MINITAB, Inc., PA, USA, 2002.
[59] R. E. Kirk, "Experimental design (2nd Ed.)," Brooks, Cole Publishing Company, 1982.

Name:
Bhekisipho Twala

Affiliation:
Director, Institute for Intelligent Systems, University of Johannesburg

Address:
P.O. Box 524, Auckland Park, Johannesburg 2006, South Africa

Brief Biographical History:
1993-2005 Received his B.A. from University of Swaziland, M.Sc. from University of Southampton, United Kingdom, Ph.D. from Open University, United Kingdom
2007 Joined the CSIR from Brunel University, United Kingdom
2010 Joined the University of Johannesburg

Main Works:
- Applied and theoretical machine learning
- Image and signal processing
- Intelligent systems

Membership in Academic Societies:
- Royal Statistical Society (RSS)
- Association of Computing Machinery (ACM)
- Chartered Institute of Transport South Africa (CITSA)
- The Institute of Electrical and Electronics Engineers (IEEE)
- International Association of Engineers (IAENG)
- South African Council for Automation and Control (SACAC)
- International Federation of Automatic Control (IFAC)