Cross-validation based *K* nearest neighbor imputation for software quality datasets: An empirical study

Jianglin Huang*, Jacky Wai Keung, Federica Sarro, Yan-Fu Li, Y.T. Yu, W.K. Chan and Hongyi Sun

J. Huang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China
E-mail: jianhuang7@cityu.edu.hk
*corresponding author. Tel: +85256013641

J.W. Keung is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China
E-mail: jacky.keung@cityu.edu.hk

F. Sarro is with the Department of Computer Science, University College London, London, UK
E-mail: f.sarro@ucl.ac.uk

Y.F. Li is with the Department of Industrial Engineering, Tsinghua University, Beijing, China
E-mail: liyanfu@tsinghua.edu.cn

Y.T. Yu is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China
E-mail: csytyu@cityu.edu.hk

W.K. Chan is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China
E-mail: wkchan@cityu.edu.hk

H. Sun is with the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China
E-mail: Sun.3333@cityu.edu.hk

Abstract: Being able to predict software quality is essential, but also it pose significant challenges in software engineering. Historical software project datasets are often being utilized together with various machine learning algorithms for fault-proneness classification. Unfortunately, the missing values in datasets have negative impacts on the estimation accuracy and therefore, could lead to inconsistent results. As a method handling missing data, *K* nearest neighbor (*K*NN) imputation gradually gains acceptance in empirical studies by its exemplary performance and simplicity. To date, researchers still call for optimized parameter setting for *K*NN imputation to further improve its performance. In the work, we develop a novel incomplete-instance based *K*NN imputation technique, which utilizes a cross-validation scheme to optimize the parameters for each missing value. An experimental assessment is conducted on eight quality datasets under various missingness scenarios. The study also compared the proposed imputation approach with mean imputation and other three *K*NN imputation approaches. The results show that our proposed approach is superior to others in general. The relatively optimal fixed parameter settings for *K*NN imputation for software quality data is also

determined. It is observed that the classification accuracy is improved or at least maintained by using our approach for missing data imputation.

Keywords: empirical software engineering estimation, *K*NN, imputation, cross-validation, missing data

Abbreviations

BMI	Bayes Multiple Imputation
CA	Classification Accuracy
CC*k*NNI	Complete-case based *K*NN Imputation
CK	Chidamber and Kemerer object-oriented metric
CVB*k*NNI	Cross-validation based *K*NN Imputation
D*k*NNI	Default *K*NN Imputation
FP	Fault-proneness
FWG*k*NNI	Feature Weighted Grey based *K*NN Imputation
G3D	GRA-based distance, $K = 3$ with Dudani adaptation based imputation
GRA	Grey Relational Analysis
GRC	Grey Relational Coefficient
GRG	Grey Relational Grade
IC*k*NNI	Incomplete-case based *K*NN Imputation
IDWM	Inverse Distance Weighted Mean
IRWM	Inverse Rank Weighted Mean
*K*NN	*K* Nearest Neighbor
LOC	Lines of Code
MAR	Missing At Random
MCAR	Missing Completely At Random
MDT	Missing Data Treatment
MEI	Mean Imputation
MI	Mutual Information
MM	Missingness Mechanism
MP	Missingness Pattern
MR	Missingness Ratio
NI	Non-ignorable
PROMISE	PRedictOr Models In Software Engineering
RMSE	Root Mean Square Error
RQ	Research Question
SEE	Software Engineering Estimation
SVM	Support Vector Machine

1. Introduction

In the domain of empirical software engineering and its related software quality estimation, researchers have devoted to predicting important quality-related variables, such as the fault count or if the fault-proneness exists, etc. Most empirical software engineering estimation builds statistical or machine learning models on historical data (Sentas and Angelis, 2006). Meanwhile, the software community has accumulated a myriad of software project quality related data for academic research, such as the PROMISE data repositories. Unfortunately, due to scarcity of software engineering data (Myrtveit et al., 2001), the significant occurrence of missing values in software datasets or known as "missingness" gradually becomes an unavoidable issue (Khoshgoftaar and Van Hulse, 2008). In addition, many properties in software engineering datasets are often indirectly measured, which leads to more frequent and complex missingness pattern to occur (Mockus, 2008).

Many estimation models cannot directly handle the missing data values; therefore, it leaves the data-preprocessing step very necessary for modern estimation process in software engineering. For example, a well-known technique called listwise deletion, had been widely adopted for handling missing values during data-preprocessing, but it potentially impairs the completeness of data and introduces undesirable biases in estimation (Huang et al., 2015). By contrast, missing data imputation methods replace missing variables by artificial estimates (Song et al., 2008); at the same time maintain the data completeness. Nowadays, more complex imputation approaches, such as random forest (Stekhoven and Bühlmann, 2012), neural network (Rey-del-Castillo and Cardeñosa, 2012), decision trees (Deb and Liew, 2016), and low-rank matrix factorization (Jing et al., 2016), have been proposed to handle the missingness issue in the applications of bioinformatics, education, ecology, energy, traffic and software engineering, etc.

When compared to mean imputation (MEI), novel approaches are still lacking popularity in software engineering estimation (SEE) (Khatibi Bardsiri et al., 2013; Kocaguneli et al., 2013a), one of which is the *K* nearest neighbor (*K*NN) imputation. The main advantage of *K*NN imputation is that it is simple and free of parametric assumptions required otherwise. It could adapt to distinct types of variables or features known to be important in estimation. *K*NN imputation had been specially applied in real-world application as a data-preprocessing step in governmental or national surveys, such as reported in Chen and Shao (2000). Its performance has

also been widely analyzed in the domain of SEE (Strike et al., 2001; Twala et al., 2005). Since most of the empirical software engineering datasets are relatively small or medium-sized, the newer robust approaches, like random forest, neural network, and low-rank matrix factorization as less than relevant. The cost of applying these sophistical approaches in practice is also unpredictable. The majority of the previous SEE studies only applied *K*NN imputation with fixed parameters when dealing with incomplete software measurement data.

Song and Shepperd (2007) once evaluated a *K*NN imputation approach with several key features classification in small-sized software effort datasets. More recently, Van Hulse and Khoshgoftaar (2014) extended the flexibility of *K*NN imputation for the software quality datasets, using incomplete-instance for missing data imputation instead of complete-instance to provide a relatively superior performance. Unfortunately, the parameter setting of the former *K*NN imputation approaches was generally predetermined for each imputation, regardless of its features or the types of missingness being imputed. While in the specialized *K*NN imputation studies, numerous efforts have been made to improve the imputation performance. The major improvement drives from two research directions (Zhang, 2012):

- Searching for the most similar *K* nearest neighbors for a given missing value;
- Final adaptation from the selected neighbors.

Both two directions are about the parameter setting in *K*NN estimator, including the distance measure, the choice of *K*, and the adaptation method. The 1st direction is the *K*NN algorithm kernel. Literature review shows that the current rule of searching the neighbors in SEE is mostly based on Minkowski distance measure. Some specialized studies of *K*NN imputation show that the grey relational analysis (GRA) based distance, is more appropriate to capture the 'nearness' (Huang and Lee, 2004). Caruana (2001) has pointed out that the *K*NN imputation could not always be superior with any possible distance measures. The choice of the *K* is subject to controversy recently. The related studies often prepopulate that the *K* from limited experience and empirical studies, other researchers argues the potential choices of *K* to be $\sqrt{N}, N > 100$ (Lall and Sharma, 1996). Where *N* is the sample size of the dataset being investigated.

The 2nd direction is computation using the selected neighbors. Missing data imputation using median/mean is a naive and effective adaptation in some cases. Using rank or distance as weight

is also popular in literature (Kocaguneli et al., 2012b). Unfortunately, there is no such a guarantee that one of these adaptations results the best option. Therefore, researchers turned to build ensembles of multi-adaptation methods to empirically find the best one under certain circumstance (Kocaguneli et al., 2012b; Kocaguneli et al., 2013b).

In this study, we focus and present a novel approach named as cross-validation based *K*NN imputation (CVB*k*NNI) to conquer the major drawback of existing *K*NN imputation approaches: an inability of adapting the parameter setting to the data. CVB*k*NNI utilizes a cross-validation scheme to search for the optimal parameter setting for estimating each missing value. CVB*k*NNI is also compared with three other *K*NN imputation approaches in the presence of artificial missingness scenarios. This empirical study:

- Introduce CVB*k*NNI, a novel approach with an adaptive parameter setting, applicable to software quality prediction and modeling. The internal design of CVB*k*NNI includes both imputation ordering and various parameters of *K*NN imputation estimator. Based on the estimators returned from the CVB*k*NN algorithm, a fixed parameter setting is discovered to be recommendable for *K*NN imputation in software quality datasets.
- Validate that the missingness scenario could be a critical factor that significantly impacts the imputation performance under certain circumstance. A thorough statistical analysis is presented to compare with the different *K*NN imputation approaches under different missingness scenarios.

In the remaining parts of the work, background and review are presented in Section 2. Section 3 introduces the CVB*k*NNI, the novel missing data imputation technique proposed in this study. The experimental design is described in Section 4. Section 5 further presents the experimental results. Section 6 discusses the known threats to validity in this empirical study. At last, the work is concluded with future work in Section 7.

2. Background

In this section, we define the terminology and provide a simple review. This section covers three aspects: an introduction to the missingness mechanisms and patterns, the review of recent

specialized *K* nearest neighbor (*K*NN) imputation studies and the missing data treatments (MDTs) research in software engineering estimation (SEE).

The missingness mechanisms (MM) and patterns (MP) explain how the missingness is summarized and classified in literature. Selecting the proper approach to deal with missing values is related to the assumption of the mechanism and pattern (Song et al., 2005). The introduction of MM, MP and ratio helps build different missingness scenarios. The performance of different MDTs could be further validated under these scenarios then. In the section of experiment design, the incomplete datasets are synthetized according to the various missingness scenarios.

The *K*NN imputation, free of data distribution assumption, is an important single hot-deck imputation technique. Popular single imputation approaches also contain mean imputation (MEI), median imputation and the ones based on stochastic regression methods, etc. Single imputation cannot tolerate the variability of characterization of the imputed values. Concisely, it is unable to provide valid confidence intervals of the imputed values. Therefore, its simultaneous accuracy as well as robustness become a concern but difficult to address adequately. As an alternative of single data imputation, multiple imputation generates many different imputed datasets and then computes the final estimation result of the complete dataset by applying appropriate adaptation strategy, which is considered more complex in its application. Novel techniques, for example, iterative imputation, gain increasing popularity in recent years, and they improve the estimation accuracy by iteratively searching for the optimal estimates until convergence.

As the specialized *K*NN imputation research has been evolved in years, yet it has been applied in contemporary SEE studies. Huang et al. (2015) has found that MEI monopolizes the imputation approaches in recent software effort estimation studies. A review of the other MDTs in SEE studies is presented at last.

2.1 Missingness mechanisms and patterns

Missingness mechanisms (MMs) and patterns (MP) make assumptions about the distribution and types of missing values (Song et al., 2008). The judgment of MM helps assess what imputation approach may be adopted (Song et al., 2005). The MM concerns if the missingness is related to

the key variable or not. It is critical as it determines how difficult handling missing values is (Song and Shepperd, 2007). There are three mechanisms (Little and Rubin, 2002): missing completely at random (MCAR), missing at random (MAR) and non-ignorable (NI). To present the MM with formal notations, assume the real-valued software data we intend to collect as $X = \{x_i\}$, $1 \le i \le N$, and X has observed and missing parts. Consider the missing parts in X have the values that are unobserved, we use the missing data indicator $M = \{m_i\}$, where $m_i = \begin{cases} 0 & \text{if } x_i \text{ is unobserved} \\ 1 & \text{if } x_i \text{ is observed} \end{cases}$, to denote the observation outcome. The missingness mechanism is characterized by the conditional probability distribution of *M* given *X*, *i.e.* $p(M \mid X, \psi)$, where ψ refers to the unknown parameters.

MCAR means there is no difference between the distribution of observed and missing values (Song et al., 2008). In other words, missingness does not depend on either observed values or missing values of X, thus $p(M \mid X, \psi) = p(M, \psi)$.

MAR means that missingness only depends on the observed values of other variable(s), not the missing ones. It does not fulfil the condition of MCAR (it must depend on at least one variable). Assuming that *m* is a potential value (vector) for M, then $\forall m \in \{1,0\}^N$ and $\forall x, y \in ^\circ{}^N$ with $H(x,y) = m : p(M = m|x) = p(M = m|y)$, where $H(x,y)$ denotes the Hamming difference vector of variables *x* and *y*, that has 0 in the positions where *x* and *y* differ and 1 in the positions where they coincide.

NI represents the situation that neither MCAR nor MAR holds (Valdiviezo and Van Aelst, 2015). Missingness only depends on the unobserved values, *i.e.* their real values. Even accounting for all the available observed information, the reason for observations being missing still depends on the unseen missingness.

Generally, there are two types of multivariate missingness patterns (MPs): **monotone** and **general** (non-monotone) (Song and Shepperd, 2007; Van Buuren, 2012). An MP is said to be monotone if an instance x_i, could be ordered such that if $x_{i,p}$ is missing then all values in x_i with $p' > p$ are missing simultaneously. It could occur in longitudinal studies. In software quality datasets, if a

major basic measure is missing, all the following derived ones will not exist. In the general pattern, missing data can occur anywhere and no special structure appears regardless of how the variables are arranged. The type of MP may affect the selection of MDTs. Strike et al. (2001) found the MDTs tend to perform worse with monotone pattern. This issue will be discussed in the experiment analysis.

Some imputation approaches cannot handle specific MMs or MPs appropriately (Song and Shepperd, 2007). MCAR could be tested by Little and Rubin (2002)'s multivariate test under certain strict conditions. Unfortunately, it is hardly applicable to validate the exact MM and MP before adopting an MDT (Song et al., 2005). Identifying MM is difficult since the prior distribution is in general unknown. Hardly it is possible to guarantee that none of MCAR, NI or MAR could exist in software quality data. Generally, the MM in real datasets is often to be either NI or MAR, while the MP often consists both general and monotone, but not always tenable (Song et al., 2005; Song et al., 2008; Strike et al., 2001). Song et al. (2008) illustrated how NI and MAR may happen in software practice. Suppose under the politic pressure, software engineers prefer not to report many high fault rates and then intend to make the values missing. While some software metrics are too difficult and time-consuming to collect, which, therefore, may cause the values missing as well. They explain how NI could happen when missingness depends on its real values. MAR could occur if only the small-sized projects were less likely to report fault rates than the large well-organized projects. It exemplifies MAR that missingness depends on the non-missing project feature: size. Therefore, this study simulates the MPs (monotone and general) and MMs (MCAR, MAR, and NI) simultaneously to conduct the experiments.

2.2 *K*NN imputation improvement

In this section, 12 former studies about specific improvements on *K*NN imputation are chronologically selected and summarized in Table 1 in terms of the imputation estimator design and the experimental data simulation (missingness injection) approaches. Note that this is not an exhaustive search on recent studies. We use the keywords combination:

(knn OR k-nn OR knni OR "nearest neighbo*") AND (imput*) AND (missing)

to search the related recent papers. Only the qualified works that concentrates on *k*NN imputation improvement for numeric variables are kept. The studies in Table 1 are simply summarized

according to the *K*NN imputation technique design and experiment design. In specific, García-Laencina et al. (2009) proposed a feature weighted distance measure based on mutual information (MI) in *K*NN imputation. Their experiment validated that both missing data imputation and classification task were improved by their technique. Hron et al. (2010) adopted the Aitchison distance in *K*NN imputation and found that it is not robust against outliers. Zhang et al. (2011) proposed a nonparametric iterative imputation algorithm (NIIA) to impute missing value and found it outperforms the other methods in general. Zhang (2011) proposed shell neighbors imputation (SNI) which fills in an incomplete instance in a given dataset by only using its left and right nearest neighbors with respect to each other. SNI was found to be better than a traditional *K*NN imputation. Zhang (2012) changed the distance measure to grey distance and found its advantage in capturing the proximity relationship. Magnussen and Tomppo (2014) calibrated *K*NN imputation with local linear regression in the context of forest science. The new technique presented improved correlation between imputation and its real value. Sahri et al. (2014) proposed FINNIM in the context of dissolved gas analysis, in which they clearly addressed two important components of imputation: ordering and estimator. Silva-Ramírez et al. (2015) combined multiplayer perceptron and *K*NN algorithms in missing data imputation and conducted their experiment on simulated datasets with different missingness patterns. Ma and Zhong (2016) proposed a correlated degree model to extract *K* nearest neighbors for imputation in the context of natural disaster science. Zhang et al. (2017) further incorporated correlation matrix in *K*NN imputation design and found its efficiency compared with the traditional *K*NN imputation.

Regarding to imputation ordering, one important component in MDT, 10 out of the 12 studies did not consider using it in *K*NN imputation. As for the *K*NN parameter: distance measure, besides the classic Euclidean distance and Manhattan distance measures, 4 out of 12 studies preferred the grey relational analysis (GRA) based similarity measure to capture the 'nearness' of neighbors. In terms of the choice of *K*, half of the studies predefined the value of *K* and the other half preferred to use overall available neighbors for adaptation. As for the adaptation methods, instead of using the mean, various methods are adopted, such as regression-based, cluster-based and Dudani weighted mean, etc. Meanwhile, less than half of the studies considered the issue of feature relevance in searching of the nearest neighbors.

As for the experiment design, the experiment data and data simulation methods are quite consistent among the studies. The UCI data, a famous machine learning data repository, has been

experimented on by half of them. The rest datasets belong to diverse professional domains, such as biology, energy, and software. Only Song and Shepperd (2007) and Van Hulse and Khoshgoftaar (2014) evaluate their new proposed *K*NN imputation approaches in the domain of empirical SEE. The missingness injection criteria for data simulation majorly consider the missingness mechanism (MM) and ratio (MR), and only Song and Shepperd (2007)'s research took into account of the missingness pattern (MP). However, only Pan et al. (2015) and Song and Shepperd (2007) empirically analyzed the impact of missingness injection on imputation performance.

To sum, for current *K*NN based missing data imputation research, it is common to see the overall methodology design is fragmented. Researchers turn to prefer different experiment evaluation criteria in studies, which, therefore, causes the corresponding technical contribution hardly justified. As for the improvement on *K*NN imputation, none of the studies systematically analyze the impacts imputation ordering in *K*NN imputation performance. There is still no common solution to select the optimized *K*NN parameters for imputation. Although researchers prefer to use various missingness scenarios to test their techniques, the significance of the impacts of the missingness scenarios are often neglected.

Two of the recent imputation approaches in Table 1, FWG*k*NN imputation (FWG*k*NNI) and IC*k*NNI, that could be repeated according to corresponding experiment design, are utilized in our experimental design as competitors to CVB*k*NNI. Pan et al. (2015) proposed a feature weighted grey based *K*NN iterative imputation (FWG*k*NNI) approach, in which they combined feature relevance and grey relational analysis (GRA) based distance measure in the estimator. MEI is used to have a preliminary estimate of the missing values. The nearest neighbors are extracted from the dataset which contains all the available instances, except the one that is to be imputed. The data is updated after each imputation iteration, and the iteration repeats until all the missing values are imputed. The capacity of FWG*k*NNI is improved compared with the 4 other competitors used in their study, including F*k*MI (Li et al., 2004), I*k*NNI (Brás and Menezes, 2007), GBNN (Huang and Lee, 2004) and G*k*NN (Zhang, 2012). Missing data injection with various MMs is also considered in their data simulation.

Van Hulse and Khoshgoftaar (2014) proposed an incomplete-case (instance) based *K*NN imputation (IC*k*NNI) in the context of software quality data, and raised the issue of missing data

in empirical SEE research once again. Instead of using all available complete instances, IC*k*NNI searches the nearest neighbor of each instance from the incomplete data. Their results showed that the complete-case based *K*NN imputation (CC*k*NNI) is far less superior than the imputation approach based on both incomplete and complete instances, *i.e*. the IC*k*NNI. The parameters of IC*k*NNI is predominated as well: Euclidean distance, $K = 5$, with mean adaptation. This paper did not consider comparing the IC*k*NNI with more imputation approaches, even the MEI.

Table 1
Major improvements of *K*NN imputation in selected studies

Imputation approach and the reference	Imputation ordering	Imputation approach				Experiment data and the simulation	
		The 3 parameters in *K*NN estimator			Feature relevance		
		Distance measure	*K*	Adaptation		Data	Data simulation
CM-*k*NN (Zhang et al., 2017)	N	Euclidean	Various	Mean	N/A	UCI and Libsvm	N/A
Novel *K*NN (Ma and Zhong, 2016)	N	GRA	By distance threshold	IDWM	N/A	A drought case	MCAR, NI, MAR
FWG*k*NN (Pan et al., 2015)	Y	GRA*	All possible neighbors	Dudani-weighted mean	Mutual information	5 UCI datasets	MCAR, NI, MAR and missingness ratio
MIMLP (Silva-Ramírez et al., 2015)	N	A similarity function	All possible neighbors	Nearest	N/A	18 datasets	MCAR, NI, MAR
IC*k*NNI (Van Hulse and Khoshgoftaar, 2014)	N	Euclidean	5	Mean	N/A	4 Software quality datasets	MCAR, NI, MAR and missingness ratio
FINNIM (Sahri et al., 2014)	Y	Manhattan	1 ~ 10	Mean	Fisher score	3 DGA datasets	N/A
*K*NN with local linear regression (Magnussen and Tomppo, 2014)	N	Euclidean	All possible neighbors	Regression	N/A	3 artificial datasets and 2 inventory datasets	Multiple sampling
G*k*NN (Zhang, 2012)	N	GRA	$\sqrt{N}$	Mean, mode	N/A	3 UCI datasets	MAR
NIIA (Zhang et al., 2011)	N	GRA	All possible neighbors	Mean, mode	Mutual information	3 UCI datasets	Missingness ratio
SNI (Zhang, 2011)	N	Euclidean	All possible neighbors	Cluster mean	N/A	6 UCI datasets	Missingness ratio
Iterative *K*NN (Hron et al., 2010)	N	Euclidean	Unknown	Geometric mean	N/A	Simulated data	Outlier ratio
MI-based *KNN* (García-Laencina et al., 2009)	N	Euclidean	2, 5	Dudani-weighted mean	Mutual information	5 UCI datasets	Missingness ratio

*GRA: grey relational analysis, which could be used to measure distance.

2.3 Studies of missing data treatment in software engineering estimation context

Missing data treatment (MDT) has been mostly discussed in the data-driven studies of social science, biology, psychology, transportation, and behavioral science (Poloczek et al., 2014; Sahri et al., 2014; Suyundikov et al., 2015). MDT is considered as an evolving area in software engineering estimation (SEE) research for less than 15 years. Less attention has been focused on

MDT methods themselves. In a more recent study, Huang et al. (2015) found that only some of the former software effort estimation studies have considered the significance of the MDTs, of which only Minku and Yao (2011) used *K*NN imputation in data-reprocessing during the estimation modeling. By contrast, Troyanskaya et al. (2001) applied *K*NN imputation in the estimation of missing DNA microarrays, and Finley et al. (2006) even explored its utility in the domain of forest science.

Empirical analysis about missingness characteristics in software quality data are even rare. Song et al. (2008) emphasized that for large-sized samples with MCAR mechanism, listwise deletion is considered appropriate, but the assumption of MCAR is ideal and less applicable in real software datasets. Additionally, if either NI or MAR exists, which is more probable, missing data imputation is relatively a better option then. However, imputation needs more thorough computational analysis (Myrtveit et al., 2001; Strike et al., 2001), and the prediction error may be introduced (Mittas and Angelis, 2010). MEI is efficient and has been involved in SEE as the most popular imputation approach; however, it will cause bias to data. MEI simply replaces the missing values with the mean of other values in the same feature.

*K*NN imputation is then used as an advanced imputation technique in SEE (Minku and Yao, 2011). Strike et al. (2001) compared and tested various parameter settings in *K*NN imputation. The settings took account of Euclidean and Manhattan distance measures. The MM is simulated from 206 real-world software datasets. The results indicated that listwise deletion is reasonable but may not provide the best performance. They called for validating more advanced imputation techniques on software engineering datasets. Myrtveit et al. (2001) evaluated the closest neighbor imputation on a real-world incomplete dataset and showed that compared to listwise deletion, *K*NN imputation is the right option only when the dataset has too much missingness. Cartwright et al. (2003) then examined MEI and *K*NN imputation for two real industrial incomplete datasets and found that *K*NN imputation provides better prediction than MEI does. Twala et al. (2005), on the other hand, recommended adopting MEI when massive missingness exists and using *K*NN imputation when sparse missingness exists. Song et al. (2005) argued that the impact of MM on imputation performance is not always that obvious. Jönsson and Wohlin (2006) examined that *K*NN imputation performs better in high dimensional incomplete datasets.

Li et al. (2007) found that more missingness in data could worsen the accuracy of *K*NN imputation. They appealed to future investigation of the impact of missingness scenarios with more distance and adaptation in *K*NN imputation. Continuously, Song et al. (2008) further confirmed that *K*NN imputation provides high accuracy. Khoshgoftaar and Van Hulse (2008) analyzed the effectiveness of various imputation approaches, including MEI, *K*NN imputation and Bayes multiple imputation (BMI), on two real software datasets. Their results indicate BMI is better than *K*NN imputation and MEI. Overall, most researchers did not consider improving *K*NN imputation in the context of SEE. Even the performance of *K*NN imputation against MEI is not consistent. As for the impact of MM or MP on imputation in software measurement datasets, few conclusions have ever reached the topic.

Based on the above discussion, the research questions (RQs) are presented as follows:

RQ1: Is *K*NN imputation on software quality data improved by using optimized and adaptive parameters?

RQ2: Does the MM or the MP have an impact on the imputation accuracy?

RQ3: Is there a fixed parameter setting of *K*NN imputation recommended for incomplete software quality data?

RQ4: Is the classification performance maintained with the imputed dataset?

The above RQs are answered in Section 5.

3. Imputation Strategy Design

This section presents the overall background used for the design of the new imputation strategy, CVB*k*NNI, including imputation ordering, estimator, and the complete algorithm. The parameters used in the study will be described in detail in Section 3.2.

3.1 Imputation ordering

Imputation ordering assigns missing values different priority levels (Sahri et al., 2014). The ordering is potentially influential to the final imputation results since each imputed value shall be included in the complete dataset iteratively for estimating the rest missing values. The criterion in this study requires the data matrix is arranged based on the missingness ratio (MR) in both

instance-row and feature-column in ascending order (Conversano and Siciliano, 2009). The missingness ratio (MR) in feature-column of one feature is defined as the number of missingness in the corresponding feature divided by the number of overall instances, *N*. While the MR in instance-row of one instance is defined as the number of missingness in the corresponding instance divided by the number of overall features, *M*. The prior ordering sequence of imputation in this work is from left to right, *i.e.* feature by feature (Van Buuren, 2012). Then the instances are re-ordered from top to bottom, according to the ascending MR in instance (row). In practice, there are small imputation sequence effects of some imputation algorithms. Evidence shows that the effects would not significantly matter (Van Buuren, 2012). Imputation ordering would maximize the information availability during each missing value imputation. The impact of imputation ordering on imputation accuracy shall be presented in the section of the experimental analysis.

3.2 Imputation estimator

The quality of *K* nearest neighbor (*K*NN) algorithm is largely dependent on the parameter tuning. There are three necessary parameters in *K*NN imputation estimator: the distance measure, the choice of *K*, and the adaptation method.

3.2.1 Distance measure

The distance measure is also referred as dissimilarity measure. Given two different instances of numeric measurements x_i and x_j, the lower distance between them, the higher similarity the represent. The distance measure used in the design of the CVB*k*NNI includes both the traditional Minkowski distance measure and transformed grey relational based measure.

- Minkowski distance

The most commonly used distance measures in former empirical software engineering estimation (SEE) studies generally belong to Minkowski distance, in which Euclidean distance and Manhattan distance gain the most popularity (Azzeh, 2012; Kocaguneli et al., 2012a; Li et al., 2009b). The Minkowski distance between x_i and x_j could be generalized as:

$$d\left(x_i, x_j\right) = \left(\left|x_{i,1} - x_{j,1}\right|^q + \left|x_{i,2} - x_{j,2}\right|^q + \cdots + \left|x_{i,p} - x_{j,p}\right|^q + \cdots + \left|x_{i,M} - x_{j,M}\right|^q\right)^{1/q} \quad (1)$$

where q is the Minkowski coefficient. Euclidean and Manhattan distance are the special cases of Minkowski distance when q = 2 or 1, respectively. Consider one historical project (instance) x_i and one rest project x_j in the same data, the weighted Euclidean/Manhattan distance between numeric features is defined as

$$d_{\text{euclidean}}(x_i, x_j) = \sqrt{\sum_{p=1}^{M} w_p \left(x_{i,p} - x_{j,p}\right)^2}, \quad (2)$$

$$d_{\text{manhattan}}(x_i, x_j) = \sum_{p=1}^{M} w_p \left|x_{i,p} - x_{j,p}\right| \quad (3),$$

where M denotes the total number of features in the data, and w_p is the normalized weight of p-th feature. In addition to Minkowski distance, researchers have also proposed other similarity/dissimilarity measures, in which grey relational analysis (GRA) based ones obtain a lot of attention in the recent literature (see Section 2.3).

- Grey relational analysis

Grey relational analysis (GRA) quantifies the impacts of different factors and the relationship among data instances. It has two fundamental measures: grey relational coefficient (GRC) and grey relational grade (GRG) (Zhang, 2012). Given instance x_l as an example, $x_l = \{x_{l,1}, x_{l,2}, x_{l,3}, ..., x_{l,M}\}$, and x_i as a random one of the rest $N-1$ instances, the GRC in p-th feature between x_l and x_i is defined as follows:

$$GRC(x_{l,p}, x_{i,p}) = \frac{\Delta_{\min} + \rho\Delta_{\max}}{\left|x_{l,p} - x_{i,p}\right| + \rho\Delta_{\max}} \quad (4),$$

where $\rho \in [0,1]$ (ρ is a distinguishing coefficient, normally, set $\rho = 0.5$ (Huang and Lee, 2004)), $\Delta_{\min} = \min_{\forall j \in [1,N] \cap j \neq l} \min_{\forall r \in [1,M]} \left|x_{l,r} - x_{j,r}\right|$, and $\Delta_{\max} = \max_{\forall j \in [1,N] \cap j \neq l} \max_{\forall r \in [1,M]} \left|x_{l,r} - x_{j,r}\right|$ (The smallest and largest value in matrix $\left|x_{l,r} - x_{j,r}\right|$). And the weighted GRG is defined as:

$$GRG(x_i, x_j) = \sum_{p=1}^{M} w_p GRC\left(x_{i,p}, x_{j,p}\right) \quad (5).$$

GRG is a similarity measure, which means that if $GRG(x_1, x_2)$ is larger than $GRG(x_1, x_3)$, the difference between x_1 and x_2 is smaller than that of x_1 and x_3. Clearly, the *GRG* takes a value between 0 and 1. Therefore, the weighted distance between x_i and x_j could be transformed to $d(x_i, x_j) = 1 - GRG(x_i, x_j)$ (Pan et al., 2015). GRA is advantageous since it measure the similarities among observations by analyzing the relational structure. Compared with Minkowski distance, the degree of 'nearness' that GRA captures will be more stable and consistent as the number of features increases. Meanwhile, each feature always has different relevance or weight in terms of calculating distance. In order to have the above-mentioned w_p during each missingness imputation, mutual information (MI) based feature relevance is considered in the process of estimating missing values in this study.

3.2.2 K

The option of *K* is highly dependent on the selected dataset, which is also critical to *K*NN imputation. Most researchers only consider *K* = 1 (Walkerden and Jeffery, 1999), some take into account of *K* = 1, 2, or 3 (Mendes et al., 2003). Li et al. (2009b) and Khatibi Bardsiri et al. (2013) recommended locating the best *K* from 1 to 5. Instead of having the same number of nearest neighbors, it is worthy to automatically find the best *K* (Kocaguneli et al., 2012a). Duda and Hart (1973) and Maier et al. (2009) suggested the upper limit of *K* being the square root of the number of instances, which limits the choices of *K*. In this study, the optimal choice of *K* is determined by 10-fold cross-validation. The upper limit of *K* is rounded to the nearest odd neighbor of $\sqrt{N}$ for the ease of computing. The range of *K* is in $\left\{ 2q+1 \middle| q \in \bullet, 0 \le q \le \frac{\sqrt{N}-1}{2} \right\}$, which contains all possible odd numbers.

3.2.3 Adaptation technique

Adaptation is the last procedure to obtain the estimate given the retrieved instances. In this study, there are five common ways of adaptations for estimating numerical values: mean, median

(Shepperd and Schofield, 1997), inverse distance weighted mean (IDWM) (Mair et al., 2000), inverse rank weighted mean (IRWM) (Kocaguneli et al., 2012b; Mendes et al., 2003) and Dudani measure (Dudani, 1976; Pan et al., 2015).

The classic measure of central tendency, mean, treats all analogies equally influential. Median is more robust to outliers than mean. IDWM makes closer neighbors have stronger influence, which is defined as:

$$\hat{y}' = \frac{\sum_{k=1}^{K} 1/\left(\delta + d\left(x_k, x'\right)\right) y_k}{\sum_{k=1}^{K} 1/\left(\delta + d\left(x_k, x'\right)\right)} \quad (6),$$

where $\hat{y}'$ is the value being estimated, $d\left(x_k, x'\right)$ is the weighted distance between x' and x_k, the k-th nearest instance of x', and δ is a small constant (δ is set to 10^{-6} in the study). Note that x' is the instance with the missing value, y_k is the corresponding feature value to x_k. IRWM, like IDWM, allows higher ranking analogies to have more influence than lower ranking ones. y_k is ranked based on the corresponding $d\left(x_k, x'\right)$ in an ascending order. The top and bottom-ranked neighbors have weights of $K / \sum_{k=1}^{K} k$ and $1 / \sum_{k=1}^{K} k$, respectively. The final IRWM estimate is defined as:

$$\hat{y}' = \frac{\sum_{k=1}^{K} \left(K - k + 1\right) y_k}{\sum_{k=1}^{K} k} \quad (7).$$

On the contrary, the Dudani measure is less used in SEE; however, it was proved to be efficiency in studies (García-Laencina et al., 2009; Pan et al., 2015). It was proposed to weigh evidence of a neighbor in *K*NN classification problems (Dudani, 1976). The weight of k-th nearest neighbor is defined in *Eq.* (8):

$$\omega_k = \begin{cases} \dfrac{\max_{\forall k \in [1,K]} d\left(x_k, x'\right) - d\left(x_k, x'\right)}{\max_{\forall k \in [1,K]} d\left(x_k, x'\right) - \min_{\forall k \in [1,K]} d\left(x_k, x'\right)}, & \max_{\forall k \in [1,K]} d\left(x_k, x'\right) \neq \min_{\forall k \in [1,K]} d\left(x_k, x'\right) \\ 1, & \max_{\forall k \in [1,K]} d\left(x_k, x'\right) = \min_{\forall k \in [1,K]} d\left(x_k, x'\right) \end{cases} \quad (8)$$

The final Dudani estimate based on the calculated weights is:

$$\hat{y}' = \frac{\sum_{k=1}^{K} \omega_k y_k}{\sum_{k=1}^{K} \omega_k} \quad (9).$$

3.3 CVB*k*NN algorithm

In this subsection, the detailed algorithm presents how the introduced components work in CVB*k*NNI in software quality data. CVB*k*NNI uses incomplete-instances for imputation. Imputing missing values from incomplete-instances could cause the results have lower bias and higher variance. Using feature relevance in distance calculation in *K*NN imputation could balance the bias-variance trade-off. This work adopts mutual information (MI) to calculate the feature relevance w_p (Li et al., 2009a). MI calculates the dependency among variables to indicate the relevance.

The entropy, $H(X)$, of a random variable *X*, measures the uncertainty of the variable. If a discrete random variable *X* has χ alphabet and the *pdf* is $p(x) = \Pr\{X = x\}, x \in \chi$, then the entropy $H(X) = -\sum_{x \in \chi} p(x) \log p(x)$ (Kullback, 1997; Pan et al., 2015). Given two random variables *X* and *Y* (*Y* has ξ alphabet and $y \in \xi$), their joint entropy *H* is defined in terms of the joint *pdf* $p(x, y)$, expressed as *Eq*. (10):

$$H(X,Y) = -\sum_{x \in \chi} \sum_{y \in \xi} p(x,y) \log p(x,y) \quad (10)$$

.

The conditional entropy calculates the resulted uncertainty on *Z* (*Z* has γ alphabet and $z \in \gamma$) given *Y*, which is:

$$H(Z \mid Y) = -\sum_{y \in \xi} \sum_{z \in \gamma} p(y,z) \log p(z \mid y) \quad (11),$$

where $p(z \mid y)$ is the conditional *pdf* of *Z* given *Y*. Furthermore, the definition of MI *I* between two variables *X* and *Y* is defined as:

$$I(X;Y) = \sum_{x \in \chi} \sum_{y \in \xi} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (12).$$

For continuous random variables, *Eq.* (12) is transformed into

$$I(X;Y) = \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \quad (13).$$

To apply MI in continuous variables, this study adopts the *mRMR* package (Peng et al., 2005). The parameter of w_p is defined as:

$$w_p = \frac{I\left(f_p; f_{\text{target}}\right)}{\sum_{p=1}^{P} I\left(f_p; f_{\text{target}}\right)} \quad (14),$$

where P, $P \leq M-1$, is the number of features in *Xtrain*, f_p, therefore, is one feature in *Xtrain* and f_{target} is *Ytrain*.

Assume that the features and instances in Table 2 are going to be rearranged by imputation ordering process. The $x_{7,2}$, *i.e.* f_2 in x_7, is going to be imputed firstly (the MR of f_2 is the minimum among f_2, f_3, f_4 and f_5, and $x_{7,2}$ is the only missing value in f_2). Then, the corresponding sub-data matrix (all available incomplete-instances) for cross-validation is filled with light and medium gray in Table 2. The sub-matrix in light gray is corresponding to *Xtrain*, and the column values in medium grey is to *Ytrain*. The cross-validation scheme searches all the possible parameter combinations to find the optimal one with the minimum validation error. Using the optimal estimator on the test instance D_{test} (filled with dark black in Table 2), together with D_{train}, obtains the estimated $\hat{x}_{7,2}$. After $x_{7,2}$ is imputed, $x_{3,3}$ is going to be imputed next (the MR of f_3 is the minimum among f_3, f_4 and f_5, and the MR of x_3 is the minimum between x_3 and x_5). This process continues until all the missing values are imputed.

Table 2
Sample data-matrix after imputation ordering (N = 7, M = 6)

ID	f_1	f_2	f_3	f_4	f_5	***Fault-proneness* (non-missingness)**
x_1	*2*	**2**	*4*	5	8	*0*
x_2	*4*	**3**	*4*	8	1	*1*
x_3	5	3	N/A	8	1	1
x_4	*1*	**1**	*5*	8	N/A	*0*

x_5	3	2	N/A	N/A	6	0
x_6	*4*	**1**	*7*	N/A	N/A	*1*
x_7	5	N/A	9	N/A	N/A	1

The detailed algorithm pseudocode is presented in **Algorithm 1**, including two parts: ordering (Line 1-5) and estimating (Line 6-20):

Algorithm 1: CVBkNNI pseudocode using Matlab notation

input : $D = [X, Y] \in \mathbb{R}^{N \times M}$
// D is the normalized incomplete data with N instances x and M features f. $X = [x_1; x_2; x_3; ...; x_N] = [f_1, f_2, f_3, ..., f_{M-1}]$ and the missing values in X are denoted as NaN. $Y = [y_1; y_2; y_3; ...; y_N] = f_M$ is the fault-proneness.
output: $D_{complete}$ // The complete D after imputation

1 Initialize KNN parameters: $K \in \{2q+1 | q \in \mathbb{N}, 0 \leq q \leq \frac{\sqrt{N}-1}{2}\}$,
$Distance \in \{Euclidean, Manhattan, GRA\}$,
$Adaptation \in \{mean, median, IDWM, IRWM, Dudani\}$;
2 $MRInstance_i \leftarrow$ MR of each instance x_i in D, $i \in [1, N]$;
3 $MRFeature_p \leftarrow$ MR of each feature f_p in D, $p \in [1, M-1]$ // No missingness in Y
4 $D_{ordered} \leftarrow$ Rearrange each f_p and then x_i in D according to MR_p and MR_i in ascending order
5 Initialize $\mathfrak{M}$, the missing data indicator matrix of $D_{ordered}$, with $m_{i,j} = 0$ if $x_{i,j} =$ NaN and 1 otherwise
// Imputation on $D_{ordered}$ from left to right, top to bottom
6 **while** *missing value* $x_{i,p} \in D_{ordered}$ $(i.e.\ m_{i,p} = 0)$ *exists* **do**
// Prepare sub-data matrix for estimating $x_{i,p}$
7 Find m_i from $\mathfrak{M}$ and set $m_{i,p} = 1$, $\mathfrak{M}_t = [m_i; m_i; m_i; ...; m_i] \in \mathbb{Z}^{N \times M}$
8 $D_{test} \leftarrow$ Complete values in x_i // Data for estimating $x_{i,p}$
9 $rowsTrain = \{a | (\mathfrak{M} \odot \mathfrak{M}_t)_a = m_i\}$ // $\odot$ is the Hadamard product
10 $colsTrain = b$ such that $m_{i,b} = 1$
11 $D_{train} = D_{ordered}[rowsTrain, colsTrain]$ // Sub-matrix of $D_{ordered}$, for training and validation
12 $Ytrain = f_p[rowsTrain]$ // Prepare Xtrain and Ytrain for 10-fold stratified CV
13 $Xtrain \leftarrow D_{train} - Y_{train}$
14 **forall** KNN estimator combination in $\{K$, MI weighted $Distance$, $Adaptation\}$ **do**
15 CV on $Xtrain$ and $Ytrain$ to find the optimal parameter combination $\{k, dis, ada\}$ whose corresponding validation error MSE is the global minimum
16 **end**
17 $\hat{x}_{i,p} \leftarrow$ Estimate $x_{i,p}$ on D_{test} using $\{k$, MI weighted $dis, ada\}$ and D_{train}
18 $D_{ordered} \leftarrow D_{ordered} + \hat{x}_{i,p}$ // Replace the missing value $x_{i,p}$ in $D_{ordered}$ with the estimate
19 **end**
20 $D_{complete} \leftarrow$ Rearrange $D_{ordered}$ to the original order of D

Steps 2-4 fulfil imputation ordering. Steps 7-13 fulfil building specific sub-data D_{train} in order to cross-validate the optimal *K*NN parameters for estimating missing value $x_{i,p}$. Steps 14-16 fulfil finding the optimal *K*NN parameters using 10-fold cross-validation. Note that in the part of

estimating, to estimate each missing value, the corresponding sub-data matrix (available-instances) is built to cross-validate the optimal *K*NN parameters. Each time the unique sub-data matrix is split into *Xtrain* and *Ytrain*, in which *Ytrain* and the target missing value(s) belong to the same feature. MI is used to measure the feature relevance between the *Xtrain* and *Ytrain* each time to automatically obtain each feature weight in *Xtrain*, *i.e.* w_p in the distance measure. As for the time complexity of the proposed CVB*k*NNI, the complexity of distance calculation in *K*NN is $O(MN)$. The total processing time in terms of sorting the distance is greater than $O(N \log N)$ in general. For each *K*NN estimator combination, the complexity of cross-validation scheme is $O(N)$. Therefore, the time complexity of imputing the whole data is $O(\alpha MN^3 \log N)$, where α is the number of *K*NN estimator combinations.

4. Experiment design

4.1 Software quality datasets

Appropriate datasets should be used to evaluate the imputation techniques. We consider the renowned tera-PROMISE Repository in the study (Menzies et al., 2016). 8 software quality datasets are selected from the repository, which are ant, arc, camel, ivy, PC5, MC2, KC3 and MW1.

The former 4 datasets, ant, arc, camel and ivy, are parts of latest Apache open source projects (Jureczko and Madeyski, 2010). The features of these four datasets are collected through Chidamber and Kemerer (CK) object-oriented code metric (Chidamber and Kemerer, 1994), one specially designed to analyze object-oriented programming languages. It groups three stages of object-oriented design: identification of classes (WMC, DIT, NOC, etc.), semantics of classes (WMC, RFC, LCOM, etc.) and relationship between classes (RFC, CBO, etc.). Similarly, all the derived measures are excluded from original data; the remaining ones of each dataset are presented in Table 4 in detail.

The last 4 datasets, MC2, PC5, KC3, and MW1, are generated from NASA C-written projects, the features of which are calculated by McCabe and Halstead's procedural metric (Halstead, 1977;

McCabe, 1976), which takes into account of program complexity and number of operators/operands. Their original data size in terms of instance count varies from around 500 to 10000. The McCabe metrics have 4 basic elements: cyclomatic complexity, design complexity, essential complexity, and Lines of Code (LOC). And the Halstead's metrics have 3 elements: base measure, derived measure and LOC. In this work, all the synthetic or derived features in the original datasets are excluded if they could be computed directly from the basic ones. The remaining features of data PC5, KC3, MC2, and MW1 are described in Table 3 in details.

In order to keep the scientific basis of empirical validation and replication of SEE studies, necessary data integrity checks require urgent intention (Shepperd et al., 2013). Besides excluding the derived measures, the following procedures are also used to select the proper instances:

1) Exclude duplicate instances.
2) Exclude the instance with implausible values, such as the values in Halstead and McCabe's metric or CK metric equal to 0 ubiquitously.
3) Exclude the instances in datasets of PC5, KC3, MC2 and MW1 that violate the referential integrity checks (Shepperd et al., 2013) on NASA software quality data.

In the end, the simple description of all the cleansed datasets are presented in Table 5.

Table 3
Feature definition for quality datasets using McCabe and Halstead's procedural metric

Metric	Features	Full name	Description
McCabe	*LOC_TOTAL*	Lines of code (LOC)	Measured according to McCabe's line counting conventions, equals to the sum of LOC_Code and Comment and LOC_Executables
	EDGE_COUNT	Control flow graph edge count	The number of edges of the graph
	v(G)	Cyclomatic complexity	Number of linearly independent paths
	ev(G)	Essential complexity	The extent to which a flow graph can be "reduced" by decomposing all the sub-flow graphs
	iv(G)	Design complexity	The v(G) of a module's reduced flow graph
	CALL_PAIRS	Call pairs	Executable calls between modules
	CONDITION_COUNT	Condition decision count	Correlates to threshold for v(G)
	DECISION_COUNT	Decision count	Correlates to threshold for v(G)
	LOC_COMMENT	lines of comment	Count of lines of comment
	LOC_BLANK	blank lines	Count of blank lines
	LOC_CODE_AND_COMMENT	Code and comment	Count of source code and comment
	PARAMETER_COUNT	Formal parameter count	Number of formal parameters

	BRANCH_COUNT	Logical branches	Branch count of the flow graph
Halstead	*UNIQ_OP*	Unique operators	Number of distinct operators
	UNIQ_OPND	Unique operand	Number of distinct operands
	TOTAL_OP	Total operator	Total number of operators
	TOTAL_OPND	Total operand	Total number of operands
	NUMBER_OF_LINES	Number of lines	End line minus the start line in the listing
	Fault-proneness	Module has/has not one or more reported defects	Fault-prone (*FP*), regarded as '1' in data, or non-fault-prone (*NFP*), regarded as '0'

Table 4
Feature definition for quality datasets using CK object-oriented metric

Metric	Features	Full name	Description
CK and its derivatives	*WMC*	Weighted methods per class	Sum of the complexities of each method in a class
	DIT	Depth of inheritance tree	Number of classes that a particular class inherits from
	NOC	Number of children	Count of immediate subclasses of a class
	CBO	Coupling between objects	Number of classes that are coupled to a class
	RFC	Response for class	Number of elements in the response set of a class
	LCOM	Lack of cohesion of methods	Number of method pairs in a class that have no common references to instance variables minus the number of method pairs that share references to instance variables
	LCOM3	Lack of cohesion in methods	Different version of LCOM suggested by Henderson-Sellers (1996), which overcomes the drawback of LCOM
	IC	Inheritance coupling	This metric provides the number of parent classes to which a given class is coupled.
	CBM	Coupling between methods	A total number of new/redefined methods to which all the inherited methods are coupled.
	AMC	Average method complexity	Average method size for each class. The size of a method is equal to the number of Java bytecodes in the method
Martin (1994)	*Ca*	Afferent couplings	Number of classes that depend upon the measured class
	Ce	Efferent couplings	Number of classes that the measured class depends upon
Bansiya and Davis (2002)	*NPM*	Number of public methods	Count of all the methods in a class that is declared as public
	DAM	Data access metric	The ratio of the number of private (protected) attributes to the total number of attributes declared in the class.
	MOA	Measure of aggregation	The extent of the part-whole relationship, realized by using attributes.
	MFA	Measure of functional abstraction	The ratio of the number of methods inherited by a class to the total number of methods accessible by the member methods of the class.
	CAM	Cohesion among methods of class	Relatedness among methods of a class based on the parameter list of the methods.
McCabe	*LOC*	Lines of code	Number of lines of code in the Java binary code of the class under investigation
	MAX_CC *AVG_CC*	Max/Avg v(G)	Number of different paths in a method plus one
	Fault-proneness	Module has/has not reported defects	Fault-prone (*FP*), regarded as '1' in data, or non-fault-prone (*NFP*), regarded as '0'

Table 5
Data description after cleaning process (Code metric, data name, number of features and instances, and *FP*/*NFP* ratio)

Metric	Dataset Name	Number of Features	*FP*/*NFP*[*]	Number of Instances
Procedural	PC5	19	258/919	1177
	KC3	19	25/111	136
	MW1	19	21/186	207
	MC2	19	20/49	69
Object-oriented	camel	21	171/625	796
	ant	21	165/504	669
	ivy	21	37/256	293
	arc	21	20/149	169

[*]The ratio of *FP*/*NFP*: ratio between the number of instances with *Fault-proneness* = 1 and that with *Fault-proneness* = 0

4.2 Missingness simulation

Missingness simulation is often used to generate various missingness scenarios to test the performance of missing data imputation techniques. In this study, three missingness mechanisms (MMs), two missingness patterns (MPs), and four missingness ratios (MRs) shall be simulated to generate 24 incomplete dataset versions. There is no missingness injected into the feature of *Fault-proneness*. MR is set to be 2.5%, 5%, 10%, and 20%, respectively. The above-mentioned three MMs (introduced in Section 2.1) are simulated after cleansing the original data. The procedures simulating each MM are presented as follows (Van Hulse and Khoshgoftaar, 2014):

- Missing Completely At Random (MCAR): Missing values are overall selected completely at random (exclude the ones from the response feature: *Fault-proneness*). Assume we have *N* instances and *M* features if we inject *MR* = 5% random missingness inside the data, there will be around $0.05 \times N \times (M-1)$ missing values in total.
- Non-ignorable (NI): A threshold set of *t* is chosen for each feature such that 75% of the instances had a value of $x_{i,p}$ less than *t*. After determining the threshold values for each feature, 40% missingness is injected into the instances with feature value(s) $x_{i,p} < t$ and the rest 60% missingness is injected into the instances with $x_{i,p} \geq t$.

- Missing At Random (MAR): It is generated by making the distribution of missing values depends on the feature of *Fault-proneness*. We implement a biased selection process where 25% missingness is injected into the *FP* instances, *i.e. Fault-proneness* equals to 1. And another 75% missingness is injected into the instances who are *NFP*, *i.e. Fault-proneness* equals to 0.

Secondly, during MM simulation on dataset instances, we use the SPSS Missing Values Analysis module to simultaneously meet the requirements of MP (Song and Shepperd, 2007). Therefore, under each MM, there shall be two scenarios corresponding to the two MPs. For the general pattern, the missingness is randomly injected into each instance. As for the monotone pattern, the missingness in each instance is mostly continuously injected. To sum, for one specific dataset, there are 24 simulated scenarios, or versions, as shown in Table 6.

Table 6
Simulated data scenarios for each dataset during experiment

MR (%)	MP	MM		
		MCAR	MAR	NI
2.5	Monotone	#1	#2	#3
	General	#4	#5	#6
5	Monotone	#7	#8	#9
	General	#10	#11	#12
10	Monotone	#13	#14	#15
	General	#16	#17	#18
20	Monotone	#19	#20	#21
	General	#22	#23	#24

4.3 Performance measure and evaluation

Error measures are fundamental to justify the prediction performance. RMSE (root mean square error) is adopted in the cross-validation scheme in CVB*k*NNI. For each true value e_i that is simulated to be missing in *D*, the corresponding imputed value is $\hat{e}_i$, then the RMSE is defined in *Eq*. (15):

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}\left(e_i - \hat{e}_i\right)^2} \quad (15),$$

where *T* denotes the total number of missing values in *D*. The relative error metrics are not considered in the study due to they are unbalanced, for example, MRE (mean of relative error)

(Foss et al., 2003). Instead, RMSE is a balanced metric and widely used in recent studies (Pan et al., 2015; Zhang, 2012; Zhang et al., 2011).

The incomplete dataset becomes a complete one after missing data imputation. The machine learning classifiers are then conducted to evaluate the impact of imputation on the performance of *Fault-proneness* classification. Four widely used classification algorithms, Discriminant analysis, *K*NN, Naive Bayes and SVM, are chosen in the study. The classification accuracy (CA) is computed via *Eq.* (16):

$$CA = \frac{1}{N}\sum_{i=1}^{N} l(FP_i, FP_i') \tag{16},$$

where N is the number of instances, FP_i' and FP_i are the classification results of the i-th instance and the corresponding real class label. $l(FP_i, FP_i') = 1$ if $FP_i = FP_i'$, and $l(FP_i, FP_i') = 0$ otherwise.

After measuring the performance, we test if the estimations of one method are significantly better than the estimations of others. To check for statistical significance, we use Wilcoxon signed-rank test. It is a non-parametric statistical hypothesis test used when comparing two related samples to assess whether their population median ranks differ (*i.e.* it is a paired difference test). Meanwhile, it is inadequate to merely show statistical significance alone; we also need to know whether the effect size is worthy of interest (Sarro et al., 2016). To assess it, we employ non-parametric Vargha-Delaney's $\hat{A}_{12}$ statistic (Arcuri and Briand, 2014). Given a performance measure X, the $\hat{A}_{12}$ statistic measures the probability that algorithm A yields better X than another algorithm B, based on the formula of $\hat{A}_{12} = \left(R_1 / \mathrm{M} - \left(\mathrm{M}+1\right)/2\right)/\mathrm{N}$, where R_1 denotes the rank sum of the first data group we are comparing, and M and N are the number of observations in the first and second data sample, respectively. If the 2 algorithms are equivalent, then $\hat{A}_{12} = 0.5$. If the first algorithm performs better than the second one, $\hat{A}_{12}$ is considered small for $0.6 \le \hat{A}_{12} < 0.7$, medium for $0.7 \le \hat{A}_{12} < 0.8$, and large for $0.8 \le \hat{A}_{12} \le 1$. The detailed experiment is provided in Section 5.1.

4.4 Experiment procedures

The experiment of the work includes 3 main tasks: simulating missingness, missing data imputation using different techniques, and the final performance evaluation. Missingness simulation is conducted on the cleansed datasets, in which the process has been discussed in Section 4.1. The simulation consists of 3 MMs (MCAR, MAR, NI), 2 MPs (Monotone, General) and 4 MRs (2.5%, 5%, 10%, 20%), 24 scenarios in total as discussed in Section 4.2. Each scenario of one dataset is replicated 30 times to reduce bias and obtain a suitable sample size. The overall experiment process is described in Fig. 1.

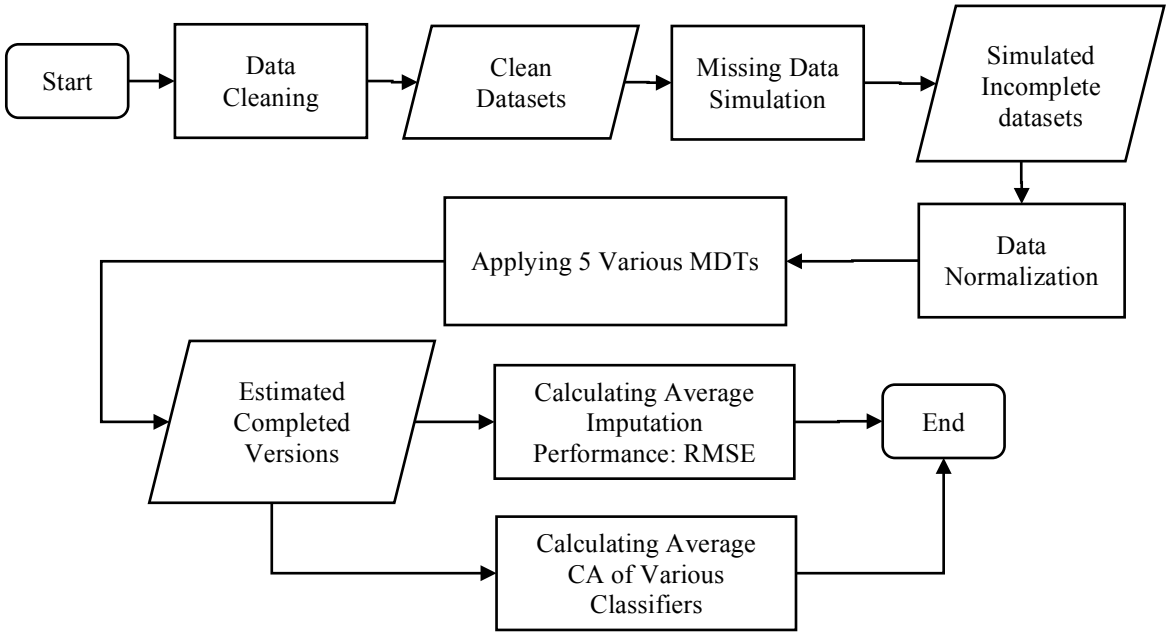

Fig. 1. The overall experiment procedures

To have the same unit for distinctive data features, it is necessary to transform the attribute values in the same range. In this work, all of the data is normalized into the interval of [0, 1] feature by feature. The [0, 1] normalization is defined as in *Eq*. (17):

$$\text{norm}_{[0,1]}\left(x_{i,j}\right) = \frac{x_{j,p} - \min_{\forall i} x_{i,p}}{\max_{\forall i} x_{i,p} - \min_{\forall i} x_{i,p}} \quad (17),$$

where $x_{i,p}$ is the p-th feature value of instance x_i, $i, j = 1, 2, ..., N$, and $p = 1, 2, ..., M$.

After normalizing all the simulated datasets, the different *K*NN imputation approaches are then used for preprocessing. The first task is the verification of the effectiveness of CVB*k*NNI.

Moreover, this study also implements three other *K*NN based imputation approaches, including FWG*k*NN (Pan et al., 2015), IC*k*NNI (Van Hulse and Khoshgoftaar, 2014), as introduced in Section 2.3, and the default version of *K*NN imputation (D*k*NNI) approach implemented by Matlab R2016b. D*k*NNI is implemented using Matlab *knnimpute*, which is capable of replacing missing data with the corresponding value from the incomplete nearest neighbor instance. According to the documentation of Matlab, D*k*NNI is based on incomplete-instance and it imputes each missing value using the closest neighbor calculated from Euclidean distance. In the meantime, MEI is also used as a benchmark imputation technique.

The imputed datasets are compared with the corresponding original complete ones to validate imputation performance. Wilcoxon signed-rank test tests whether the overall prediction performance of CVB*k*NNI is significantly better than the rest four ones. Meanwhile, this work also uses Wilcoxon signed-rank test to find if there exists a significant difference in terms of imputation among diverse scenarios.

For the adopted quality data, the target class for classification is *Fault-proneness*. Researchers argue that the imputed complete datasets should also be reliable and workable to be used for other purpose (Sahri et al., 2014). In empirical software quality research, data imputation may also serve the further *Fault-proneness* classification; therefore, the classification performance from imputed data should not be worse than that from the original data. At this stage, the four commonly used ML classifiers (Discriminant analysis, *K*NN, Naive Bayes and SVM) are implemented on the estimated complete datasets to test the performance of used imputation approaches, as a necessary data-preprocessing step, on classification tasks.

5. Experiment results and analysis

In this section, the empirical results of various imputation approaches are fully presented. The comparison between CVB*k*NNI and other imputation approaches is discussed then via statistical tests. Later, a detailed discussion about CVB*k*NNI and its inner adaptive parameter setting is presented as well.

5.1 Overall imputation performance

Table 7 presents the overall RMSEs for each dataset under different missingness scenarios. The datasets are ordered by nature and size. All the best estimation results are marked in green, the second-best ones are marked in blue, while the worst results are in red. It is obvious that CVB*k*NNI surpasses the other four imputation approaches under each scenario regardless of the missingness mechanism (MM), pattern (MP) or ratio (MR), especially when the size of the dataset is relatively large (See Table 5). The second-best imputation approach then strongly depends on the MP. FWG*k*NNI performs better when the MP is general; while IC*k*NNI performs relatively better when the MP is monotone. However, FWG*k*NNI, compared with IC*k*NNI, is relatively more robust since when MP is general, IC*k*NNI mostly performs the worst, even worse than the benchmark approach mean imputation (MEI). Some exceptions happen when the percentage of missing values is relatively small, such as dataset KC3 and MC2. The IC*k*NNI was established to be better than complete-instance *K* nearest neighbor (*K*NN) imputation; however, its performance in the software quality datasets shows that it could be even worse in imputation capacity than the benchmark imputation approaches, the default D*k*NNI and MEI. Meanwhile, in dataset camel and ant, the performance of IC*k*NNI under monotone pattern is not strictly negatively correlated with MR. It may due to the impacts of outliers in the dataset.

Table 7 also presents the Wilcoxon signed-rank test results together with the corresponding $\hat{A}_{12}$ effect size (see Table 7 footnote) to compare the statistical significance and effect size of the improvements over the other imputation approaches due to CVB*k*NNI. For example, the dataset camel, as shown in Table 7, under general pattern, MCAR mechanism and 2.5% missingness ratio, the 30 RMSEs of CVB*k*NNI (Avg: 0.088) are significantly less than the 30 RMSEs of FWG*k*NNI (Avg: 0.113), at the significance level of 0.01. Similarly, under monotone pattern, all else are equal, the 30 RMSEs of CVB*k*NNI (Avg: 0.093) are significantly less than the 30 RMSEs of IC*k*NNI (Avg: 0.111). The test results further confirm the imputation excellency of CVB*k*NNI since, in most cases, the RMSEs of CVB*k*NNI are significantly less than those of FWG*k*NNI, IC*k*NNI, D*k*NNI and MEI. Some reasonable exceptions exist in the small-sized datasets or under monotone pattern. As for $\hat{A}_{12}$ effect size shown in Table 7 (presented in different brackets), large effect size $0.8 \le \hat{A}_{12} \le 1$ dominates the results mostly, especially in the

4 object-oriented datasets, which means CVB*k*NNI overall yields better performance. Table 8 further organizes all the results of effect size in detailed counts and ratios. For each dataset, we count the number of large, medium, small and rest effect size of CVB*k*NNI vs. the other imputation approaches under all missingness scenarios. All the effect size calculated is at least 0.5. For dataset MC2, a relatively smaller one, the corresponding effect size is generally small. But this phenomenon does not happen in small-sized dataset arc. For large-sized dataset PC5, the effect size is merely medium in general.

To further intuitively present the imputation accuracy, Fig. 2 to 9 present the boxplots of the corresponding RMSE results. For example, in Fig. 2, the first sub-boxplot presents the RMSE results of the 5 imputation approaches on the 30 simulated versions of dataset camel under general MP and MCAR mechanism at MR = 5%. To save space, only the boxplots of RMSEs of the large-sized datasets: camel, ant, PC5 and MW1, at MR = 5% and 20% are presented. The results shown in the boxplots are consistent with the findings in Table 7. The overall performance of CVB*k*NNI basically answers to the RQ1, that setting adaptive parameters for estimating each missing value could largely improve *K*NN imputation performance.

Another important issue of performance, time, is also tested in the experiment. The complicated strategy of CVB*k*NNI causes the algorithm to be time-consuming, but it also provides better accuracy. Use datasets of camel, ivy, PC5 and MW1 as examples, the imputation algorithm running time is summarized in Table 9. We run the algorithms on an Intel Core i7-4770 3.40GHz CPU with 8GB memory, Windows 7 64-bit system and Matlab R2016b software. Since the algorithm running time under different MMs and MPs is relatively unchanged given a specific MR, Table 9 provides the average running time of the 5 imputation algorithms under 3 MMs and 2 MPs. Compared to the other four algorithms, CVB*k*NNI indeed cost lots of time to proceed, but it is still acceptable. The datasets of camel and PC5 are the largest ones in the experiment. Consider under the worst-case MR = 20%, there are in total 3184 missing values for camel data and 4237 ones for PC5 data, the imputation time of CVB*k*NNI is still within 3mins

Table 7
Overall average RMSE results of all datasets under various scenarios[1]

Data	MP	MR (%)	Imputation Approaches and MMs														
			MCAR					MAR					NI				
			CVB*k*NNI	FWG*k*NNI	IC*k*NNI	D*k*NNI	MEI	CVB*k*NNI	FWG*k*NNI	IC*k*NNI	D*k*NNI	MEI	CVB*k*NNI	FWG*k*NNI	IC*k*NNI	D*k*NNI	MEI
camel	General	2.5	0.088	0.113**	0.209**	0.197**	0.197**	0.080	0.115**	0.203**	0.194**	0.194**	0.114	0.149**	0.247**	0.230**	0.231**
		5	0.094	0.122**	0.213**	0.199**	0.199**	0.088	0.118**	0.209**	0.194**	0.195**	0.118	0.157**	0.251**	0.235**	0.236**
		10	0.105	0.128**	0.213**	0.197**	0.197**	0.107	0.130**	0.215**	0.198**	0.198**	0.134	0.171**	0.255**	0.241**	0.242**
		20	0.124	0.138**	0.215**	0.197**	0.198**	0.125	0.140**	0.216**	0.198**	0.199**	0.169	0.195**	0.260**	0.250**	0.251**
	Monotone	2.5	0.093	0.124**	[0.111**]	0.133**	0.223**	0.107	0.139**	[0.120**]	0.137**	0.232**	0.118	0.164**	[0.144**]	0.163**	0.256**
		5	0.109	0.135**	0.171**	0.205**	0.226**	0.118	0.146**	0.173**	0.206**	0.233**	0.121	0.162**	[0.142**]	0.164**	0.254**
		10	0.112	0.144**	0.176**	0.214**	0.231**	0.114	0.145**	[0.127**]	0.149**	0.230**	0.149	0.183**	0.192**	0.226**	0.270**
		20	0.120	0.148**	[0.135**]	0.154**	0.230**	0.122	0.149**	0.183**	0.212**	0.230**	0.194	0.233**	0.236**	0.258**	0.292**
ant	General	2.5	0.095	0.120**	0.203**	0.190**	0.195**	0.099	[0.120**]	0.194**	0.176**	0.180**	0.123	0.160**	0.230**	0.221**	0.227**
		5	0.100	0.124**	0.204**	0.190**	0.195**	0.098	0.123**	0.200**	0.182**	0.187**	0.127	0.164**	0.234**	0.225**	0.230**
		10	0.110	0.129**	0.204**	0.188**	0.192**	0.107	0.127**	0.201**	0.184**	0.187**	0.139	0.170**	0.232**	0.227**	0.233**
		20	0.129	0.139**	0.206**	0.188**	0.192**	0.129	0.138**	0.204**	0.187**	0.191**	0.171	0.185**	0.235**	0.235**	0.241**
	Monotone	2.5	0.115	0.145**	[0.129**]	0.155**	0.214**	0.123	0.155	(0.131**)	0.165**	0.228**	0.140	0.175**	[0.166**]	0.189**	0.250**
		5	0.126	0.158**	0.165**	0.198**	0.222**	0.126	0.160**	0.164**	0.193**	0.226**	0.135	0.170**	[0.162**]	0.184**	0.247**
		10	0.125	0.157**	0.161**	0.190**	0.223**	0.123	0.158**	0.135**	0.162**	0.225**	0.155	0.186**	0.191**	0.218**	0.260**
		20	0.129	0.159**	0.144**	0.164**	0.223**	0.125	0.160**	0.176**	0.202**	0.223**	0.192	0.222**	0.235**	0.251**	0.280**
ivy	General	2.5	0.090	0.125**	0.205**	0.196**	0.198**	0.088	0.114**	0.190**	0.181**	0.187**	0.107	0.171**	0.241**	0.237**	0.241**
		5	0.090	0.126**	0.200**	0.191**	0.194**	0.094	0.120**	0.197**	0.184**	0.189**	0.121	0.176**	0.252**	0.242**	0.245**
		10	0.109	0.134**	0.205**	0.192**	0.195**	0.107	0.134**	0.203**	0.189**	0.194**	0.147	0.186**	0.255**	0.245**	0.249**
		20	0.131	0.143**	0.204**	0.192**	0.195**	0.135	0.148**	0.208**	0.193**	0.199**	0.186	0.208**	0.260**	0.255**	0.259**
	Monotone	2.5	0.131	0.165**	[0.156**]	0.187**	0.224**	0.113	0.163**	[0.130**]	0.157**	0.228**	0.131	0.198**	0.162**	0.194**	0.277**
		5	0.118	0.156**	(0.126)	0.155**	0.225**	0.118	0.164**	0.163**	0.199**	0.231**	0.136	0.201**	0.190**	0.233**	0.279**
		10	0.121	0.164**	0.164**	0.198**	0.229**	0.127	0.170**	0.165**	0.198**	0.233**	0.156	0.207**	[0.170**]	0.189**	0.284**
		20	0.122	0.164**	0.136**	0.156**	0.228**	0.128	0.177**	0.184**	0.209**	0.233**	0.213	0.249**	0.241**	0.259**	0.306**
arc	General	2.5	0.096	[0.136**]	0.216**	0.208**	0.216**	0.104	[0.136**]	0.209**	0.205**	0.212**	0.112	0.196**	0.269**	0.270**	0.273**
		5	0.086	0.131**	0.220**	0.208**	0.211**	0.100	0.143**	0.226**	0.211**	0.216**	0.138	0.198**	0.273**	0.269**	0.272**
		10	0.108	0.145**	0.228**	0.213**	0.215**	0.122	0.155**	0.234**	0.217**	0.221**	0.152	0.207**	0.281**	0.277**	0.281**
		20	0.148	[0.156*]	0.231**	0.216**	0.218**	0.154	[0.163**]	0.234**	0.216**	0.220**	0.214	0.243**	0.292**	0.292**	0.297**
	Monotone	2.5	0.123	0.195**	0.168*	0.209**	0.243**	0.133	0.192**	[0.169*]	0.203**	0.239**	0.117	0.194**	0.180**	0.212**	0.298**
		5	0.128	0.200**	(0.148**)	[0.164**]	0.244**	0.129	0.196**	0.169**	0.197**	0.243**	0.135	0.204**	0.166**	0.175**	0.307**
		10	0.135	0.200**	0.170**	0.201**	0.250**	0.125	0.199**	0.175**	0.212**	0.253**	0.173	0.246**	0.209**	0.229**	0.328**
		20	0.132	0.195**	0.179**	0.204**	0.251**	0.138	0.213**	0.177**	0.190**	0.253**	0.240	0.283**	(0.249)	(0.259*)	0.332**
PC5	General	2.5	0.063	[0.080**]	0.093**	[0.088**]	[0.089**]	0.062	(0.076**)	[0.087**]	[0.083**]	[0.085**]	0.089	[0.121**]	0.128**	0.128**	0.130**
		5	0.064	[0.082**]	0.094**	0.088**	0.089**	0.065	[0.081**]	0.092**	[0.086**]	0.087**	0.096	[0.123**]	0.134**	0.131**	0.133**
		10	0.065	0.080**	0.092**	0.086**	0.087**	0.069	0.082**	0.093**	0.088**	0.089**	0.103	[0.125**]	0.132**	0.129**	0.131**
		20	0.071	[0.079**]	0.090**	0.084**	0.085**	0.073	[0.080**]	0.093**	0.086**	0.087**	0.118	(0.127**)	[0.132**]	[0.130**]	[0.132**]
	Monotone	2.5	0.057	{0.061}	{0.057}	{0.057}	(0.070**)	0.054	{0.058}	{0.053}	[0.069*]	(0.063*)	0.087	(0.112**)	{0.091}	(0.097)	(0.118*)
		5	0.062	{0.071*}	(0.071**)	[0.083**]	(0.076**)	0.065	(0.078**)	{0.068}	[0.080**]	[0.081**]	0.082	(0.108**)	(0.089**)	(0.095**)	[0.117**]
		10	0.060	(0.071**)	{0.065**}	[0.075**]	[0.078**]	0.061	(0.075**)	(0.067**)	0.079**	[0.080**]	0.089	[0.109**]	{0.091}	{0.097**}	[0.115**]
		20	0.063	[0.075**]	(0.066)	0.075**	0.079**	0.064	[0.078**]	(0.074**)	[0.079**]	[0.079**]	0.116	(0.128**)	(0.130**)	(0.130**)	[0.132**]
MW1	General	2.5	0.083	0.132**	0.183**	0.170**	0.179**	0.092	0.139**	0.205**	0.185**	0.200**	0.117	0.176**	0.207**	0.207**	0.228**
		5	0.092	0.132**	0.188**	0.171**	0.180**	0.104	0.139**	0.204**	0.185**	0.194**	0.126	0.186**	0.234**	0.221**	0.233**

		10	0.106	0.130**	0.179**	0.165**	0.172**	0.119	0.146**	0.206**	0.187**	0.197**	0.153	0.198**	0.237**	0.226**	0.237**
		20	0.119	0.136**	0.180**	0.165**	0.171**	0.129	0.150**	0.211**	0.188**	0.194**	0.194	0.214**	0.241**	0.235**	0.246**
	Monotone	2.5	0.105	0.140**	(0.115**)	0.138**	0.164**	0.122	[0.162**]	{0.130*}	[0.153**]	0.197**	0.133	[0.186**]	(0.151**)	(0.164**)	0.222**
		5	0.114	0.148**	(0.125*)	0.144**	0.170**	0.130	0.167**	[0.145**]	0.165*	0.205**	0.142	0.192**	(0.154**)	[0.175**]	0.231**
		10	0.122	0.161**	0.141**	0.158**	0.182**	0.124	0.168**	[0.139**]	0.158**	0.201**	0.168	0.205**	{0.173}	[0.195**]	0.239**
		20	0.124	0.159**	(0.130*)	0.150**	0.180**	0.129	0.193**	0.161**	0.175**	0.196**	0.202	0.238**	{0.205}	(0.213**)	0.262**
KC3	General	2.5	0.113	0.165**	0.183**	0.176**	0.195**	0.111	0.159**	0.185**	0.182**	0.192**	0.141	0.221**	0.216**	0.229**	0.258**
		5	0.113	0.162**	0.212**	0.194**	0.194**	0.118	0.170**	0.218**	0.203**	0.202**	0.166	0.239**	0.271**	0.262**	0.269**
		10	0.115	0.160**	0.209**	0.188**	0.189**	0.121	0.165**	0.215**	0.194**	0.195**	0.187	0.248**	0.270**	0.266**	0.270**
		20	0.134	0.161**	0.216**	0.186**	0.188**	0.131	0.166**	0.217**	0.191**	0.193**	0.227	0.264**	0.278**	0.274**	0.278**
	Monotone	2.5	0.114	(0.153)	{0.117*}	(0.146*)	0.171**	0.135	(0.171**)	{0.138}	(0.151)	0.194**	0.154	(0.201**)	{0.161**}	[0.202**]	0.236**
		5	0.133	[0.184**]	{0.142**}	[0.171**]	0.197**	0.140	0.197**	(0.149**)	0.181**	0.212**	0.167	0.224**	{0.169}	[0.205**]	0.253**
		10	0.136	0.183**	(0.142**)	0.167**	0.198**	0.137	0.175**	{0.141}	0.168**	0.192**	0.192	0.260**	{0.195}	(0.217**)	0.280**
		20	0.140	0.206**	0.154**	0.175**	0.217**	0.139	0.196**	[0.149**]	0.176**	0.209**	0.205	0.273**	{0.208*}	(0.218**)	0.288**
MC2	General	2.5	0.103	{0.114*}	(0.123*)	(0.140**)	[0.136**]	0.092	{0.105*}	(0.111**)	[0.126**]	[0.140**]	0.184	(0.221**)	(0.200*)	{0.199}	[0.243**]
		5	0.107	(0.124**)	0.150**	0.145**	0.148**	0.100	(0.115*)	[0.128**]	[0.142**]	[0.146**]	0.205	0.232**	[0.227*]	(0.224)	0.252**
		10	0.123	[0.138*]	0.171**	0.160**	0.161**	0.126	(0.135*)	0.168**	0.159**	0.158**	0.220	(0.236**)	[0.255**]	[0.249**]	0.253**
		20	0.145	(0.149)	0.179**	0.168**	0.170**	0.145	{0.146}	0.180**	0.166**	0.167**	0.246	{0.248}	[0.264**]	(0.257*)	[0.262**]
	Monotone	2.5	0.144	{0.192}	{0.163}	{0.150}	(0.221**)	0.132	{0.155**}	{0.142}	(0.143)	(0.170**)	0.168	(0.195*)	{0.177}	{0.175}	[0.220**]
		5	0.138	{0.164*}	{0.144}	(0.162)	(0.183**)	0.148	{0.168*}	{0.150}	{0.151}	(0.180**)	0.171	(0.196**)	(0.178)	[0.185]	[0.217**]
		10	0.125	(0.149**)	{0.131}	[0.145**]	[0.167**]	0.131	{0.147*}	{0.132}	[0.165*]	[0.163**]	0.185	(0.205**)	{0.185}	[0.198]	[0.220**]
		20	0.134	(0.150**)	(0.139*)	[0.150**]	[0.164**]	0.127	(0.143**)	{0.131*}	0.159**	[0.155**]	0.221	(0.232**)	(0.227)	{0.231*}	[0.244**]

[1]The minimum RMSEs in each condition are in green; the maximum RMSEs are in red; while the 2nd smallest RMSEs are marked in blue.

{}/()/[]: Curly brackets for effect size $0.5 \le \hat{A}_{12} < 0.6$, parentheses for small effect size $0.6 \le \hat{A}_{12} < 0.7$, square brackets for medium effect size $0.7 \le \hat{A}_{12} < 0.8$, and no brackets for large effect size $0.8 \le \hat{A}_{12} \le 1$.

**Left-tail Wilcoxon signed-rank test, significant at the level of 0.01; *Significant at the level of 0.05.

Table 8

Counts and ratios of Vargha-Delaney's $\hat{A}_{12}$ statistic from the overall RMSE results

Data	Imputation Approaches	Counts and Ratios of Vargha-Delaney's $\hat{A}_{12}$ Effect Size			
		$0.5 \le \hat{A}_{12} < 0.6$	$0.6 \le \hat{A}_{12} < 0.7$	$0.7 \le \hat{A}_{12} < 0.8$	$0.8 \le \hat{A}_{12} \le 1$
camel	CVB*k*NNI vs. FWG*k*NNI	0/24[1]	0/24	0/24	24/24
	CVB*k*NNI vs. IC*k*NNI	0/24	0/24	6/24	18/24
	CVB*k*NNI vs. D*k*NNI	0/24	0/24	0/24	24/24
	CVB*k*NNI vs. MEI	0/24	0/24	0/24	24/24
ant	CVB*k*NNI vs. FWG*k*NNI	0/24	0/24	1/24	23/24
	CVB*k*NNI vs. IC*k*NNI	0/24	1/24	3/24	20/24
	CVB*k*NNI vs. D*k*NNI	0/24	0/24	0/24	24/24
	CVB*k*NNI vs. MEI	0/24	0/24	0/24	24/24
ivy	CVB*k*NNI vs. FWG*k*NNI	0/24	0/24	0/24	24/24
	CVB*k*NNI vs. IC*k*NNI	0/24	1/24	3/24	20/24
	CVB*k*NNI vs. D*k*NNI	0/24	0/24	0/24	24/24
	CVB*k*NNI vs. MEI	0/24	0/24	0/24	24/24
arc	CVB*k*NNI vs. FWG*k*NNI	0/24	0/24	4/24	20/24
	CVB*k*NNI vs. IC*k*NNI	0/24	2/24	1/24	21/24
	CVB*k*NNI vs. D*k*NNI	0/24	1/24	1/24	22/24
	CVB*k*NNI vs. MEI	0/24	0/24	0/24	24/24
PC5	CVB*k*NNI vs. FWG*k*NNI	3/24	8/24	11/24	2/24
	CVB*k*NNI vs. IC*k*NNI	6/24	6/24	2/24	10/24
	CVB*k*NNI vs. D*k*NNI	2/24	3/24	9/24	10/24
	CVB*k*NNI vs. MEI	0/24	4/24	10/24	10/24
MW1	CVB*k*NNI vs. FWG*k*NNI	0/24	0/24	2/24	22/24
	CVB*k*NNI vs. IC*k*NNI	3/24	5/24	2/24	14/24
	CVB*k*NNI vs. D*k*NNI	0/24	2/24	3/24	19/24
	CVB*k*NNI vs. MEI	0/24	0/24	0/24	24/24
KC3	CVB*k*NNI vs. FWG*k*NNI	0/24	3/24	1/24	20/24
	CVB*k*NNI vs. IC*k*NNI	8/24	2/24	1/24	13/24
	CVB*k*NNI vs. D*k*NNI	0/24	4/24	3/24	17/24
	CVB*k*NNI vs. MEI	0/24	0/24	0/24	24/24
MC2	CVB*k*NNI vs. FWG*k*NNI	9/24	13/24	1/24	1/24
	CVB*k*NNI vs. IC*k*NNI	9/24	6/24	4/24	5/24
	CVB*k*NNI vs. D*k*NNI	5/24	5/24	8/24	6/24
	CVB*k*NNI vs. MEI	0/24	4/24	13/24	7/24

[1]There are 2 MPs, 3 MMs and 4 MRs, in total, 24 scenarios

Table 9

Average algorithm running time in seconds on 4 selected datasets

Data	MR (%)	Avg. Algorithm Running Time (in seconds)				
		CVB*k*NNI	FWG*k*NNI	IC*k*NNI	D*k*NNI	MEI
camel	2.5	50.8839	0.7549	0.0082	0.0091	0.0004
	5	86.1633	0.9416	0.0110	0.0143	0.0004
	10	129.7130	1.3375	0.0165	0.0224	0.0004
	20	167.8360	1.8559	0.0282	0.0385	0.0004
ivy	2.5	37.0678	0.8673	0.0141	0.0094	0.0002
	5	48.9862	0.7437	0.0137	0.0076	0.0002
	10	88.9723	1.0780	0.0112	0.0138	0.0002
	20	111.3102	1.3277	0.0170	0.0194	0.0002
PC5	2.5	45.1913	0.6650	0.0087	0.0107	0.0004
	5	74.1314	0.8746	0.0126	0.0186	0.0004
	10	103.4838	1.1381	0.0172	0.0223	0.0004
	20	147.9999	1.5946	0.0290	0.0409	0.0004

KC3	2.5	11.7655	0.3755	0.0027	0.0023	0.0002
	5	21.0672	0.4802	0.0036	0.0040	0.0002
	10	33.9020	0.7103	0.0066	0.0093	0.0002
	20	51.5934	0.8911	0.0101	0.0127	0.0002

From the results showing in boxplots, the median values under NI mechanism are always slightly larger than that under MCAR or MAR mechanism. In Table 7, the average RMSEs under monotone pattern are generally large than that under general pattern within the same dataset. This section also uses Wilcoxon signed-rank test to answer RQ2: if the MM, or MP indeed has a significant impact on the imputation results. Table 10 and Table 11 summarize the comparison results. The comparison between each pair of MMs is presented in Table 10. The five imputation approaches used in the study (CVB*k*NNI, FWG*k*NNI, IC*k*NNI, D*k*NNI and MEI) are denoted as 1, 2, 3, 4 and 5 accordingly. As shown in Table 10, in dataset camel, under general pattern and 2.5% missingness ratio, all the five imputation approaches perform significantly different between MCAR and NI, as well as between MAR and NI; however, none of which performs significantly different between MCAR and MAR. Table 10 shows various imputation approaches perform similarly under MCAR or MAR; while the significant difference exists when mechanism is NI. The significance may increase as the MR increases as well.

The comparison in terms of the MP is presented in Table 11. The difference in object-oriented datasets is more significant than that in procedural datasets. When MR increases in small-sized datasets, the difference is even more clear. Therefore, the impact of MP may depend on the data. The performance of IC*k*NNI is highly influenced by the MP, which is consistent with the findings in Table 7.

Table 10
The comparison between each pair of MMs at the significance level of 0.05[1]

Data	MP	MM	MR = 2.5%		MR = 5%		MR = 10%		MR = 20%	
			MAR	NI	MAR	NI	MAR	NI	MAR	NI
camel	General	MCAR	N/A	12345	N/A	12345	N/A	12345	N/A	12345
		MAR	-	12345	-	12345	-	12345	-	12345
	Monotone	MCAR	2	12345	2	12345	34	12345	34	12345
		MAR	-	245	-	345	-	12345	-	12345
ant	General	MCAR	45	12345	45	12345	N/A	12345	N/A	12345
		MAR	-	12345	-	12345	-	12345	-	12345
	Monotone	MCAR	N/A	12345	N/A	5	34	12345	34	12345
		MAR	-	3	-	5	-	12345	-	12345
ivy	General	MCAR	N/A	12345	N/A	12345	N/A	12345	N/A	12345
		MAR	-	12345	-	12345	-	12345	-	12345
	Monotone	MCAR	34	25	34	12345	N/A	125	234	12345

		MAR	-	2345	-	12345	-	125	-	12345
arc	General	MCAR	N/A	2345	N/A	12345	1	12345	N/A	12345
		MAR	-	2345	-	12345	-	12345	-	12345
	Monotone	MCAR	N/A	5	4	5	N/A	12345	N/A	12345
		MAR	-	5	-	5	-	12345	-	12345
PC5	General	MCAR	N/A	12345	N/A	12345	N/A	12345	N/A	12345
		MAR	-	2345	-	12345	-	12345	-	12345
	Monotone	MCAR	N/A	4	N/A	N/A	N/A	1235	N/A	12345
		MAR	-	N/A	-	N/A	-	12	-	12345
MW1	General	MCAR	N/A	1245	N/A	12345	12345	12345	12345	12345
		MAR	-	1245	-	12345	-	12345	-	12345
	Monotone	MCAR	N/A	N/A	145	1235	N/A	12345	12345	12345
		MAR	-	N/A	-	2	-	12345	-	12345
KC3	General	MCAR	N/A	1245	N/A	12345	N/A	12345	5	12345
		MAR	-	12345	-	12345	-	12345	-	12345
	Monotone	MCAR	N/A	N/A	N/A	N/A	N/A	12345	N/A	12345
		MAR	-	N/A	-	N/A	-	12345	-	12345
MC2	General	MCAR	1	1235	3	12345	N/A	12345	N/A	12345
		MAR	-	1235	-	12345	-	12345	-	12345
	Monotone	MCAR	N/A	N/A	N/A	N/A	N/A	1234	N/A	12345
		MAR	-	N/A	-	N/A	-	1235	-	12345

[1]1 = CVB*k*NNI, 2 = FWG*k*NNI, 3 = IC*k*NNI, 4 = D*k*NNI, 5 = MEI

Table 11
The comparison between general pattern and monotone pattern at the significance level of 0.05[1]

Data	MM	General and Monotone			
		MR = 2.5%	MR = 5%	MR = 10%	MR = 20%
camel	MCAR	345	235	2345	2345
	MAR	1245	125	345	345
	NI	345	345	12345	1235
ant	MCAR	2345	1235	1235	2345
	MAR	1235	1235	12345	2345
	NI	345	345	1235	125
ivy	MCAR	1235	12345	1235	12345
	MAR	12345	1235	12345	2345
	NI	1345	1235	2345	1235
arc	MCAR	23	12345	1235	1235
	MAR	1235	1235	235	12345
	NI	34	345	12345	2345
PC5	MCAR	34	3	2345	1345
	MAR	235	N/A	3	13
	NI	N/A	34	34	N/A
MW1	MCAR	13	13	123	234
	MAR	13	123	234	234
	NI	34	34	34	234
KC3	MCAR	3	3	134	235
	MAR	3	13	134	2345
	NI	N/A	34	34	34
MC2	MCAR	N/A	N/A	3	3
	MAR	N/A	1	N/A	3

	NI	N/A	4	12345	N/A

[1]1 = CVB*k*NNI, 2 = FWG*k*NNI, 3 = IC*k*NNI, 4 = D*k*NNI, 5 = MEI

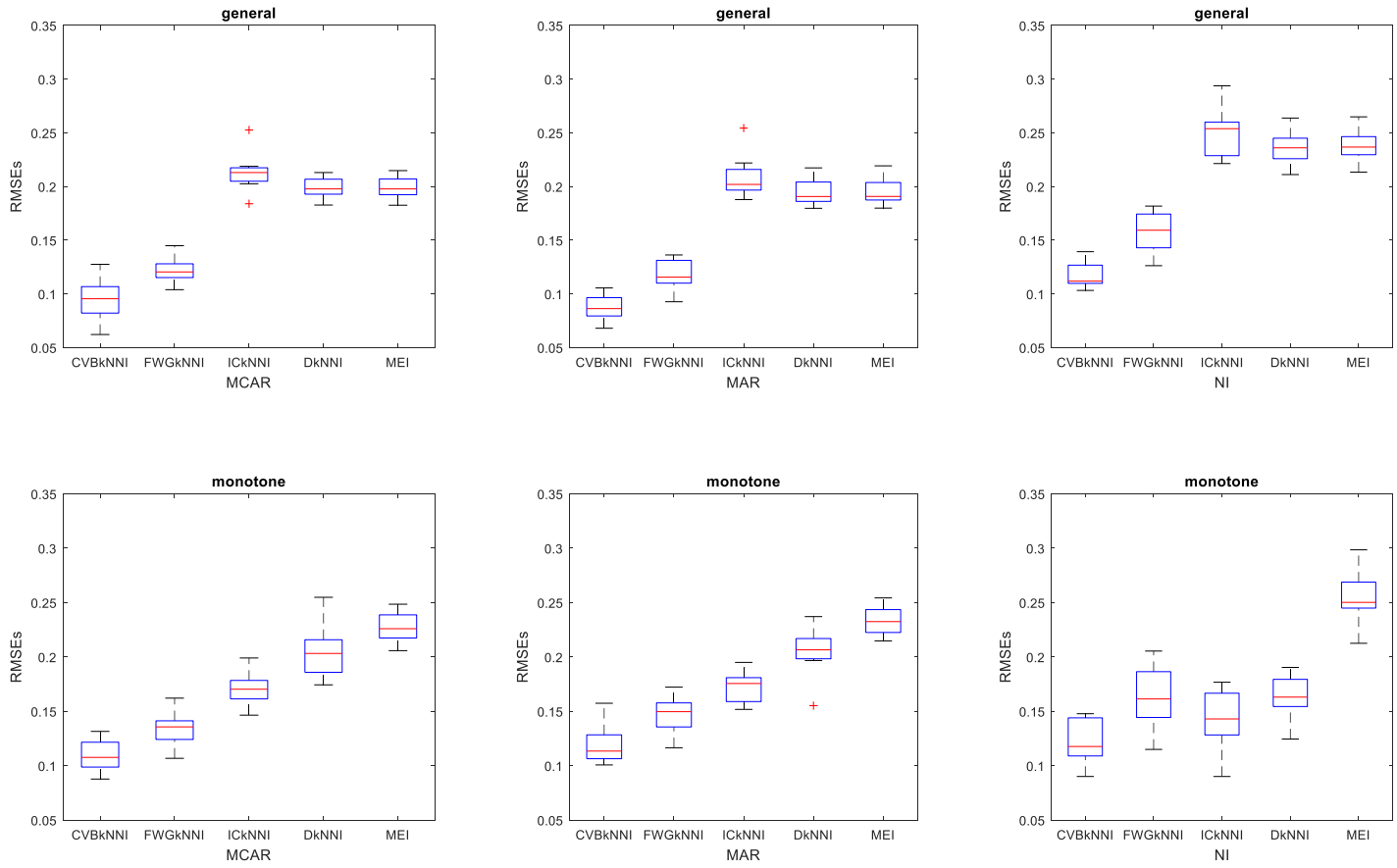

Fig. 2 The RMSEs of data camel at MR = 5%, range in [0.05, 0.35]

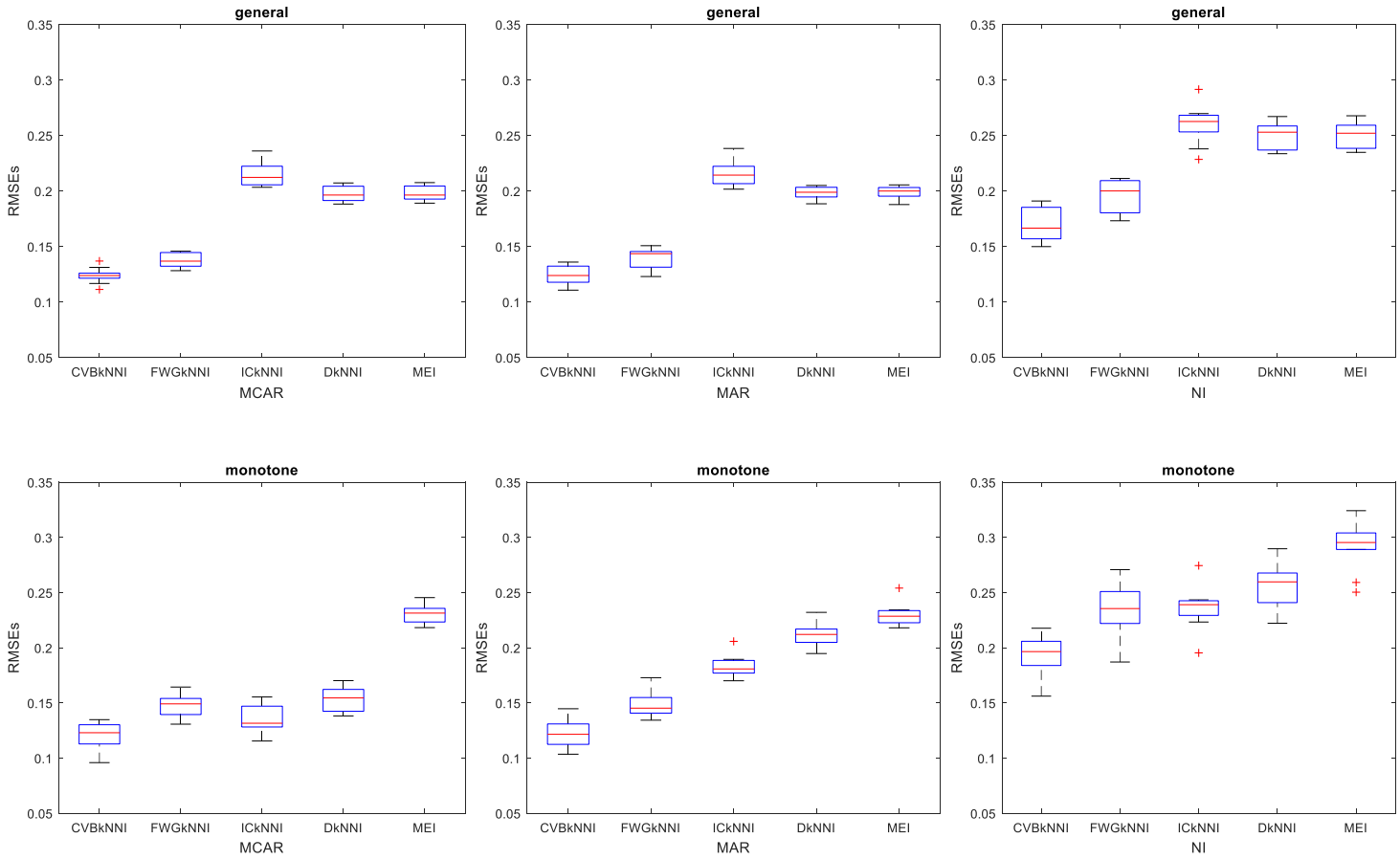

Fig. 3. The imputation RMSEs of data camel at MR = 20%, range in [0.05, 0.35]

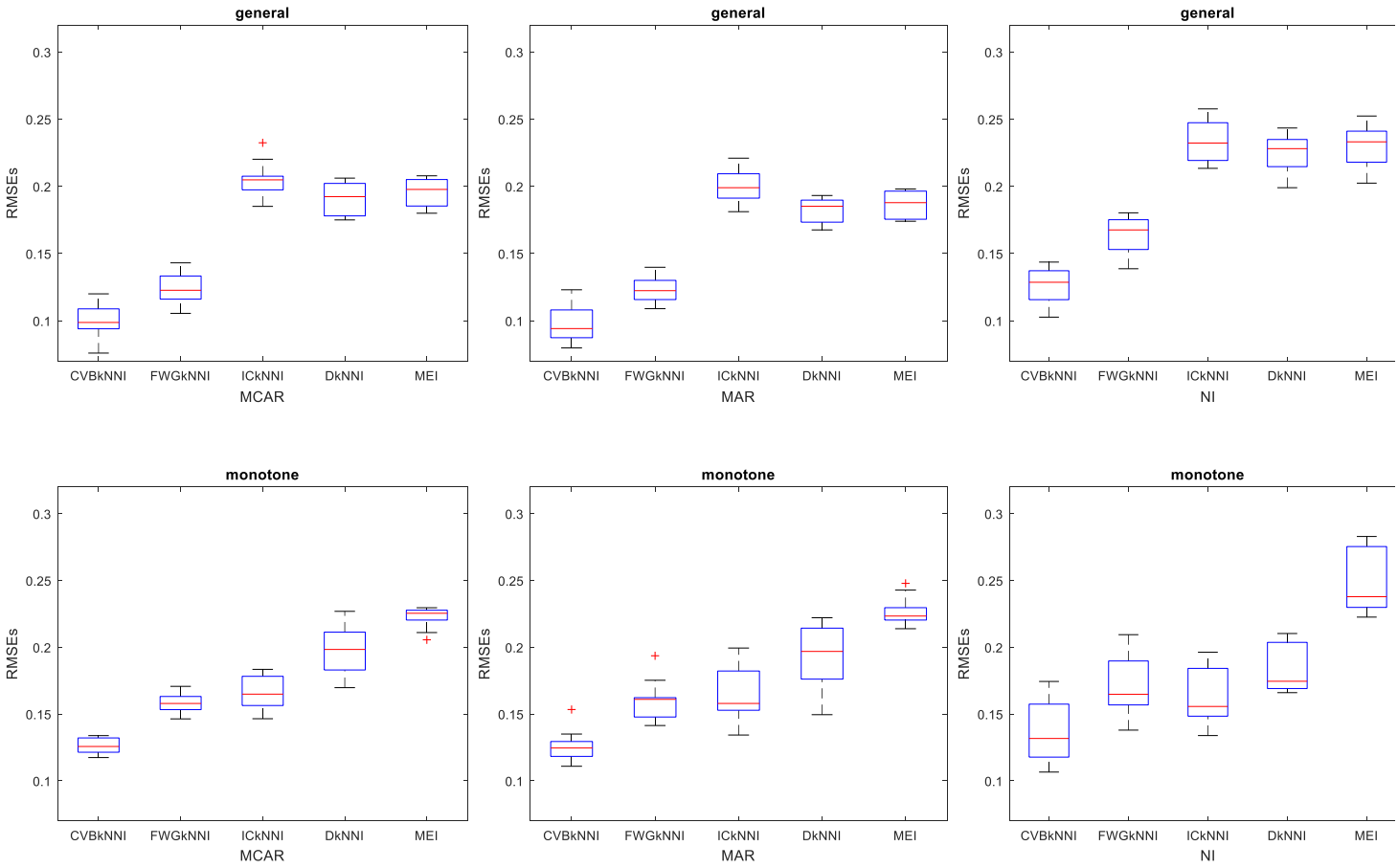

Fig. 4 The imputation RMSEs of data ant at MR = 5%, range in [0.07, 0.32]

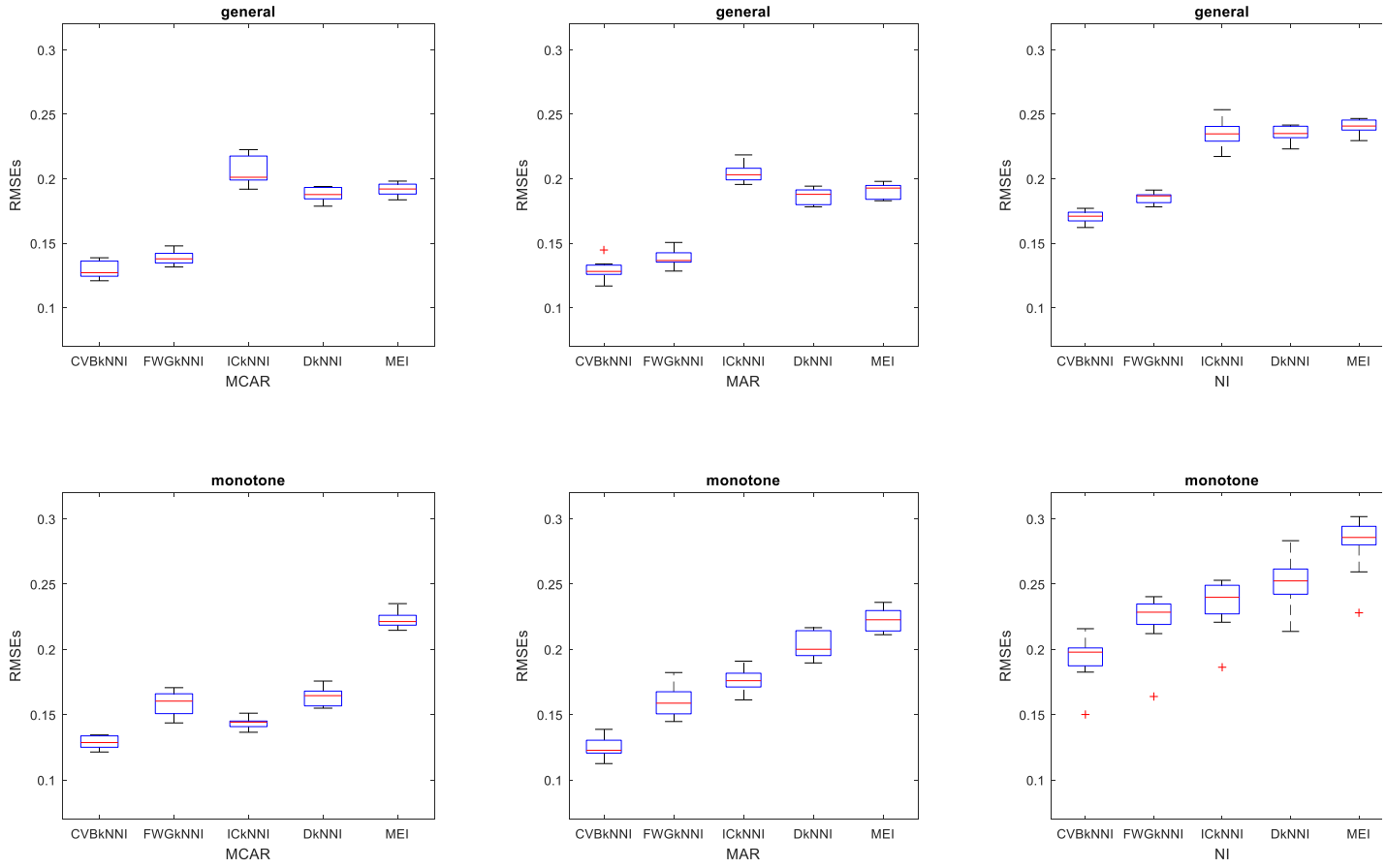

Fig. 5 The imputation RMSEs of data ant at MR = 20%, range in [0.07, 0.32]

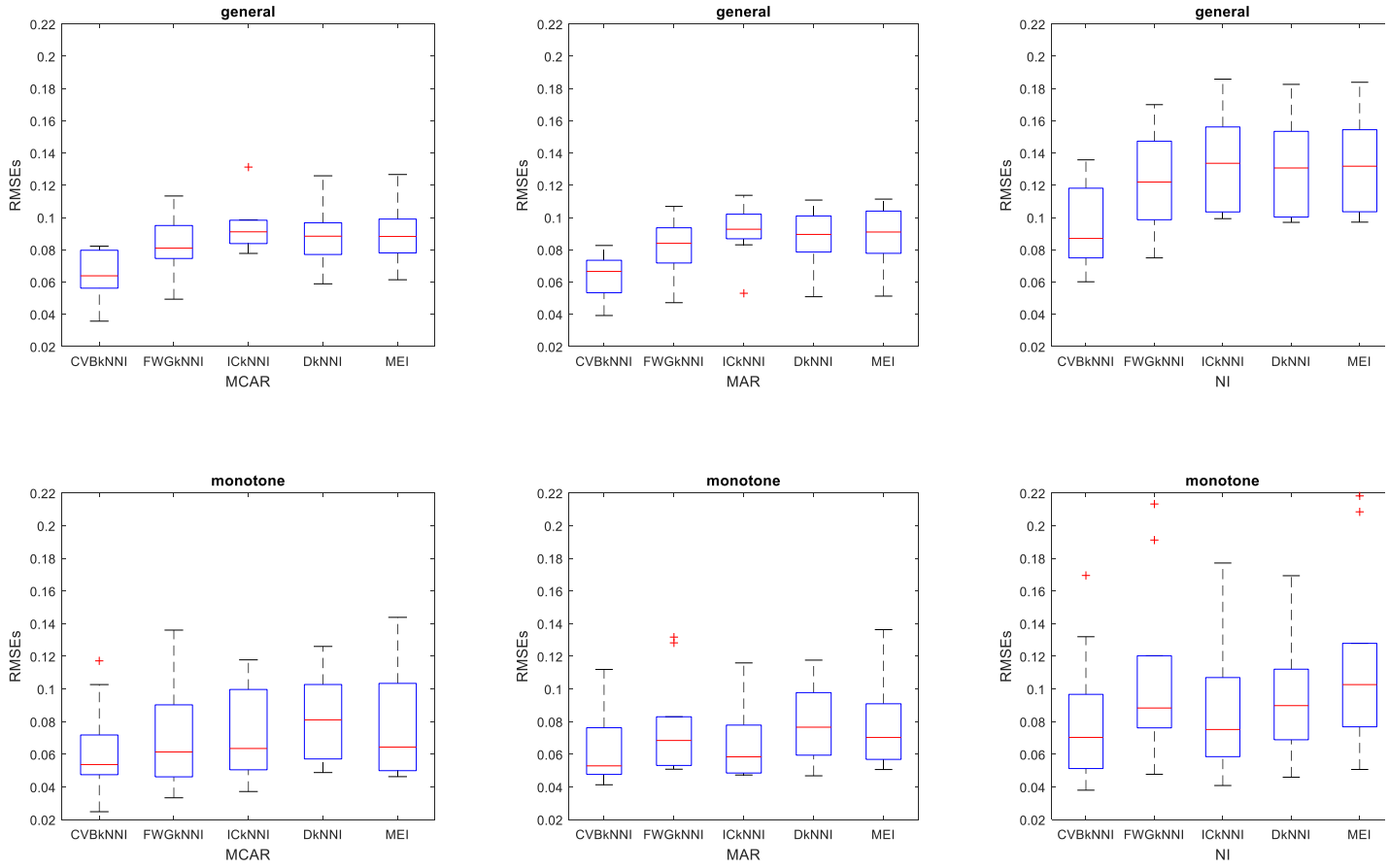

Fig. 6. The imputation RMSEs of data PC5 at MR = 5%, range in [0.02, 0.22]

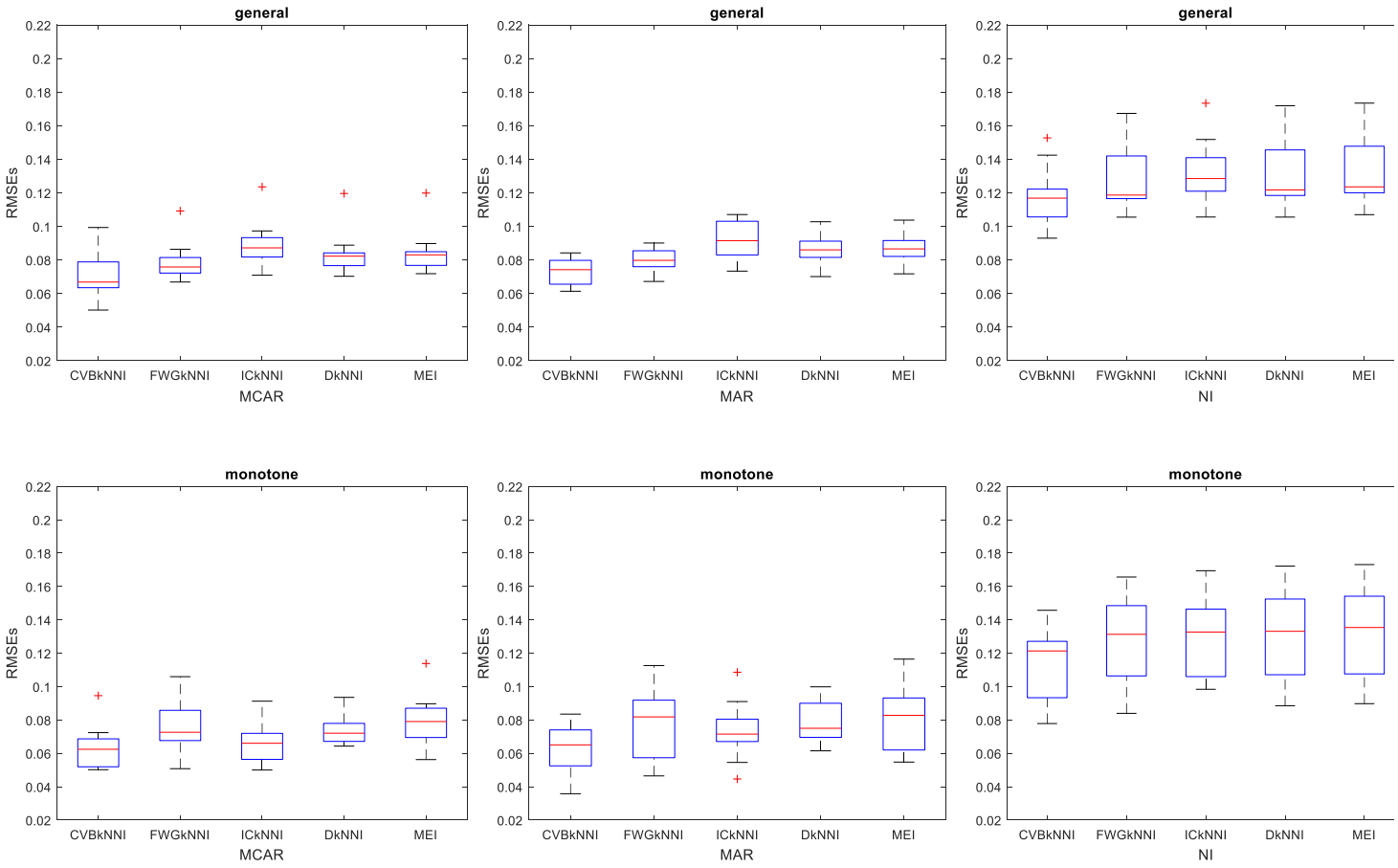

Fig. 7. The imputation RMSEs of data PC5 at MR = 20%, range in [0.02, 0.22]

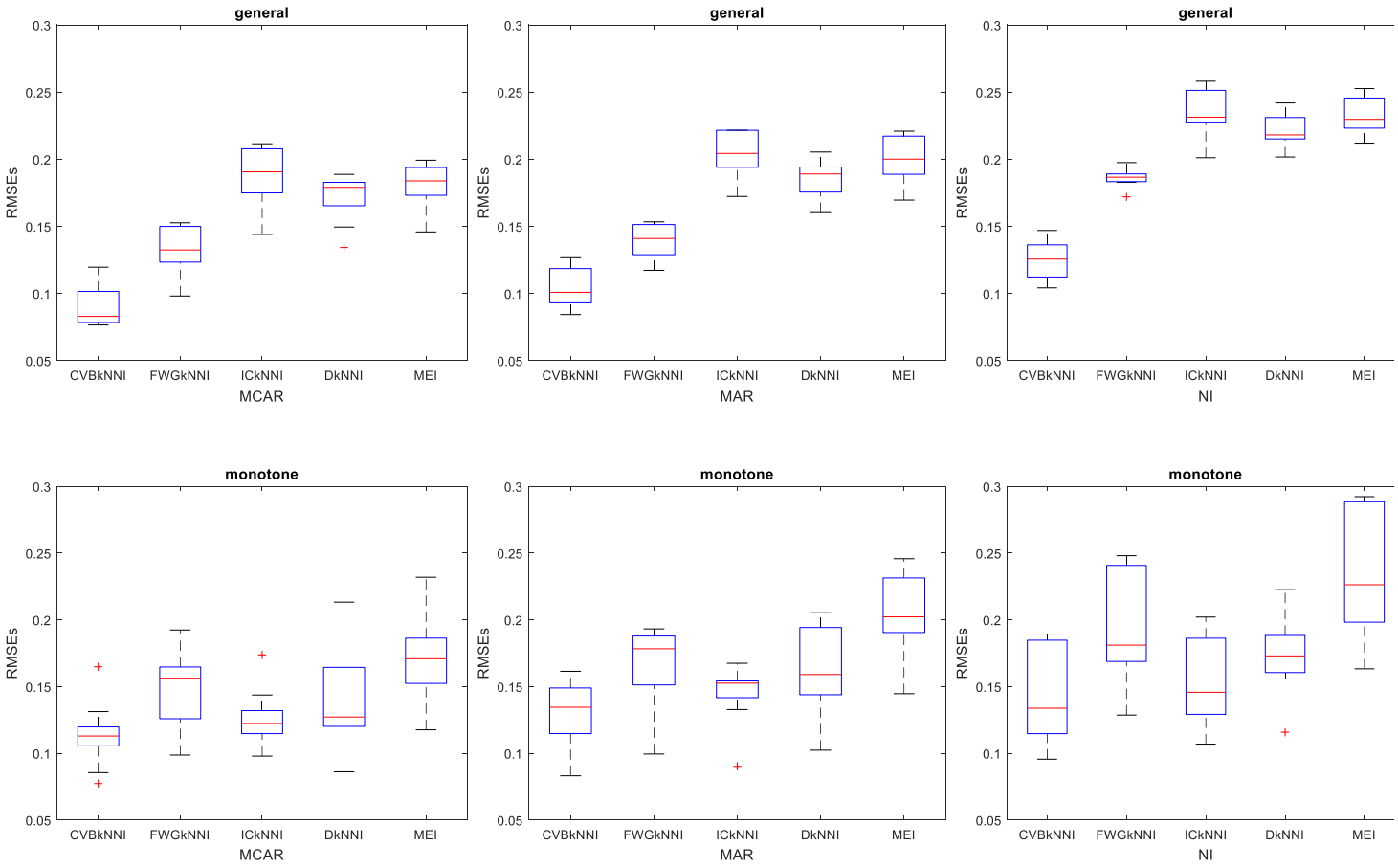

Fig. 8 The imputation RMSEs of data MW1 at MR = 5%, range in [0.05, 0.3]

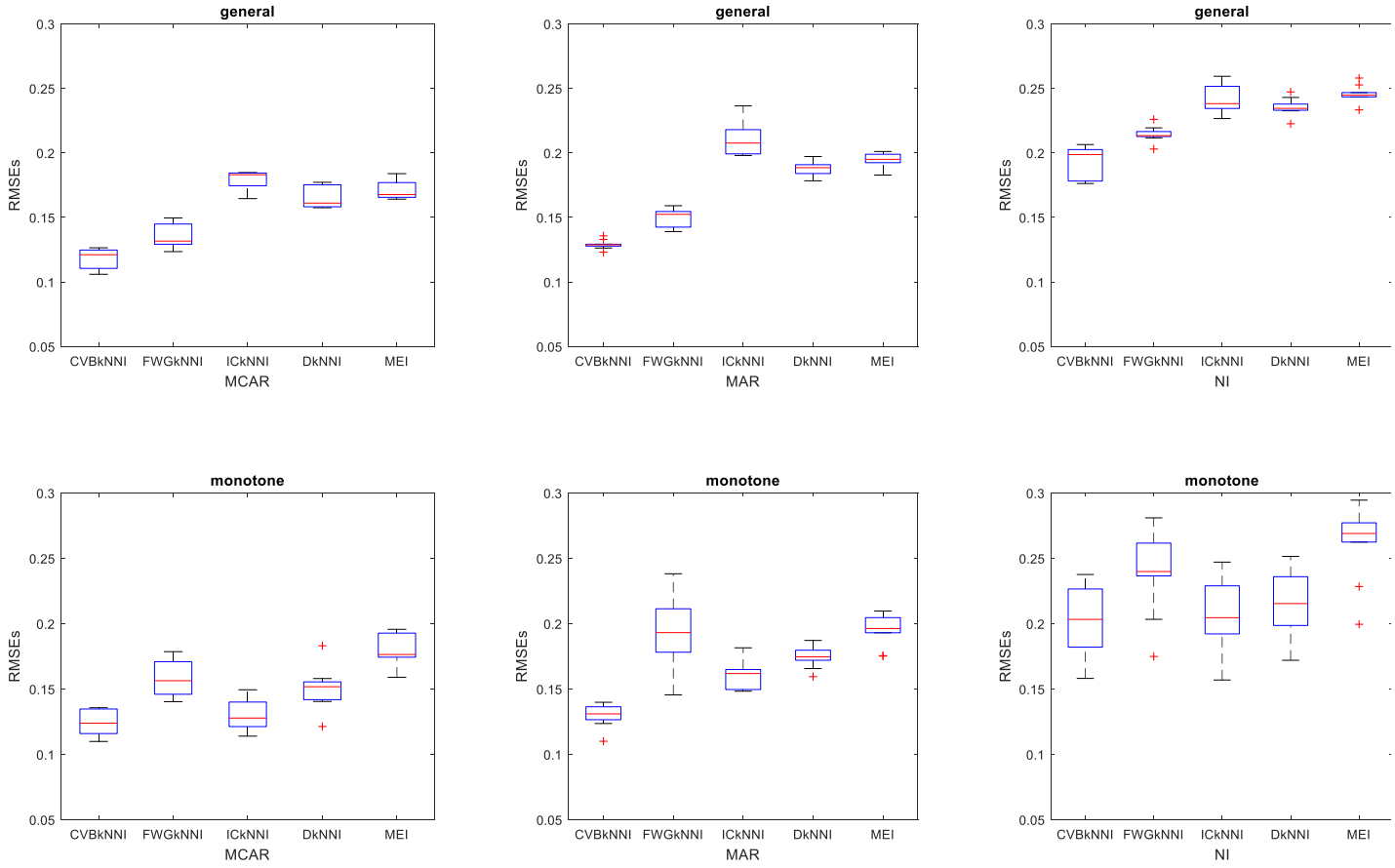

Fig. 9 The imputation RMSEs of data MW1 at MR = 20%, range in [0.05, 0.3]

5.2 The impact of feature relevance and imputation ordering

This section and the following one focus on empirically analyzing the estimator of CVB*k*NNI. The two components used in CVB*k*NNI, MI-based feature relevance, and MR-based imputation ordering, are both inherited from former empirical research evidence. This section aims at verifying the impact of the two components on imputation accuracy. Table 12 gives an example of a comparison in terms of various configurations of CVB*k*NNI, *i.e.* with (w/) or without (w/o) MI-based feature relevance, and w/ or w/o MR-based imputation ordering, at MR = 10%. To save space, Table 12 only lists the results of four datasets with MR = 10%. When MR = 2.5%, or 5% or 20%, the results are like those in Table 12. The configuration with both MI-based feature relevance and imputation ordering is the CVB*k*NNI used in this study (Marked in bold in Table 12).

Left-tail Wilcoxon signed-rank test is also used to test if the RMSEs are reduced with imputation ordering, as well as if the RMSEs are reduced with feature relevance considered in measuring distance. For example, in dataset camel, under the general MP and MCAR, the average RMSEs of the CVB*k*NNI is measured as 0.105 (Also shown in Table 7), which is significantly less than that without feature relevance (0.112) at the significant level of 0.05. Similarly, if imputation ordering is excluded in the CVB*k*NNI, the corresponding RMSEs (Avg: 0.106) after the CVB*k*NNI with feature relevance are significantly less than those (Avg: 0.113) without feature relevance. Therefore, from the results, if the component of feature relevance is excluded, the imputation accuracy of the CVB*k*NNI is reduced in most cases. However, if feature relevance is included, the average imputation performance of the CVB*k*NNI with ordering is slightly better than that without ordering, especially when the MM is NI. Imputation ordering may not have an overall significant impact on the performance of the CVB*k*NNI, but at least, it reduces the average RMSE in general.

Table 12
A comparison in terms of different configurations of CVB*k*NNI at MR = 10%

Data	MP	MM	w/ MI-based feature relevance		w/o MI-based feature relevance	
			w/ imputation ordering[1]	w/o imputation ordering	w/ imputation ordering	w/o imputation ordering
camel	General	MCAR	**0.105**[2]	0.106	0.112*	0.113*
		MAR	**0.107**	**0.107**	0.120*	0.120*
		NI	**0.134**	0.154*	**0.134**	0.154*
	Monotone	MCAR	**0.112**	0.113	0.114	0.115
		MAR	**0.114**	0.116	0.121*	0.127*
		NI	**0.149**	0.151*	**0.149**	0.151*
ant	General	MCAR	0.110	**0.109**	0.115*	0.116*
		MAR	**0.107**	**0.107**	0.118*	0.118*
		NI	**0.139**	0.141*	0.150*	0.153*
	Monotone	MCAR	**0.125**	**0.125**	0.138*	0.135*
		MAR	**0.123**	0.125	0.138*	0.132*
		NI	**0.155**	0.157	0.161*	0.164*
PC5	General	MCAR	**0.065**	0.068*	0.082*	0.080*
		MAR	**0.069**	**0.069**	**0.069**	**0.069**
		NI	**0.103**	0.104	0.104	0.107
	Monotone	MCAR	**0.060**	0.061	0.067*	0.069*
		MAR	**0.061**	**0.061**	0.078*	0.075*
		NI	**0.089**	0.098*	0.090	0.099*
MW1	General	MCAR	**0.106**	**0.106**	**0.106**	0.108
		MAR	**0.119**	0.120	0.125*	0.123
		NI	**0.153**	0.159*	**0.153**	0.159*
	Monotone	MCAR	**0.122**	0.124	**0.122**	**0.122**
		MAR	**0.124**	0.126	0.129*	0.129*
		NI	**0.168**	0.182*	0.183*	0.184*

[1]CVB*k*NNI configuration.

[2]The results in terms of RMSEs are consistent with those in Table 7.
[*]Left-tail Wilcoxon signed-rank test at the significant level of 0.05

5.3 Parameter setting in CVB*k*NNI

Moreover, the selected parameter combinations in CVB*k*NNI for estimating missing values in a dataset are summarized as well. Table 13 lists the mostly selected distance measure, *K* and adaptation method under each missingness scenario in the 30-time replicated camel, ant, PC5 and MW1 datasets.

The results in Table 13 show that GRA-based distance measure obviously takes the majority. For relatively small-sized MW1 data, the use of GRA-based distance measure is slightly overwhelmed by Manhattan distance. For the choice of *K*, most of the imputation parameters prefer $K = 3$. As for the adaptation method, Dudani measure is more popular than both mean and IDWM. Neither IRWM nor median is selected. Fig. 10 and Fig. 11 further present the histograms of the distribution of each utilized parameter. The results on figures are consistent with Table 13. A combination of parameter setting (GRA-based distance measure, $K = 3$, with Dudani adaptation measure) is overall the most frequent for the imputation. However, there is still no guarantee that any combination is overall the best solution.

Table 13
The mostly selected parameter setting under each scenario

Data examples	MP	MR (%)	MM								
			MCAR			MAR			NI		
			Distance	*K*	Adaptation	Distance	*K*	Adaptation	Distance	*K*	Adaptation
camel	General	2.5	GRA[1]	3	Dudani	GRA	3	Dudani	GRA	3	Dudani
		5	GRA	3	Dudani	GRA	3	Dudani	GRA	5	Dudani
		10	GRA	3	Dudani	GRA	3	Dudani	GRA	3	Dudani
		20	GRA	3	Dudani	GRA	3	Dudani	GRA	3	Dudani
	Monotone	2.5	GRA	5	IDWM	GRA	7	IDWM	GRA	5	Dudani
		5	GRA	5	IDWM	GRA	5	IDWM	GRA	3	Dudani
		10	GRA	7	IDWM	GRA	5	IDWM	GRA	3	Dudani
		20	GRA	5	IDWM	GRA	3	Dudani	GRA	3	Dudani
ant	General	2.5	GRA	5	Dudani	GRA	5	Dudani	GRA	5	Dudani
		5	GRA	3	Dudani	GRA	3	Dudani	GRA	5	Dudani
		10	GRA	3	Dudani	GRA	3	Dudani	GRA	3	Dudani
		20	GRA	3	Dudani	GRA	3	Mean	GRA	3	Dudani
	Monotone	2.5	GRA	19	IDWM	GRA	11	IDWM	GRA	5	Dudani
		5	GRA	19	Dudani	GRA	17	IDWM	GRA	5	Dudani
		10	GRA	17	Dudani	GRA	7	IDWM	GRA	5	IDWM
		20	GRA	7	IDWM	GRA	15	Dudani	GRA	3	Dudani

PC5	General	2.5	GRA	3	Dudani	GRA	3	Dudani	GRA	3	Mean
		5	GRA	3	Mean	GRA	3	Dudani	GRA	3	Mean
		10	GRA	3	Dudani	GRA	3	Dudani	GRA	3	Dudani
		20	GRA	3	Mean	GRA	3	Mean	GRA	3	Mean
	Monotone	2.5	GRA	3	Mean	GRA	3	Dudani	GRA	3	Dudani
		5	GRA	3	Dudani	GRA	3	Mean	GRA	3	Dudani
		10	GRA	3	Mean	GRA	3	Mean	GRA	3	Mean
		20	GRA	3	Dudani	GRA	1	Mean	GRA	3	Mean
MW1	General	2.5	Manhattan	3	Dudani	Manhattan	3	Dudani	Manhattan	3	Dudani
		5	Manhattan	3	Dudani	GRA	3	Dudani	Manhattan	3	Dudani
		10	Manhattan	3	Dudani	GRA	3	Dudani	Manhattan	3	Dudani
		20	GRA	3	Dudani	Euclidean	3	Dudani	GRA	3	Dudani
	Monotone	2.5	Manhattan	5	Dudani	Manhattan	3	Dudani	Manhattan	3	Dudani
		5	Manhattan	5	Dudani	Manhattan	5	Dudani	Manhattan	3	Dudani
		10	GRA	3	Dudani	Manhattan	3	Dudani	Manhattan	3	Dudani
		20	Manhattan	3	IDWM	Manhattan	3	IDWM	GRA	3	Dudani

[1]GRA denotes the distance measure of $d\left(x_i, x_j\right) = 1 - GRG\left(x_i, x_j\right)$, as introduced in Section 3.2.1

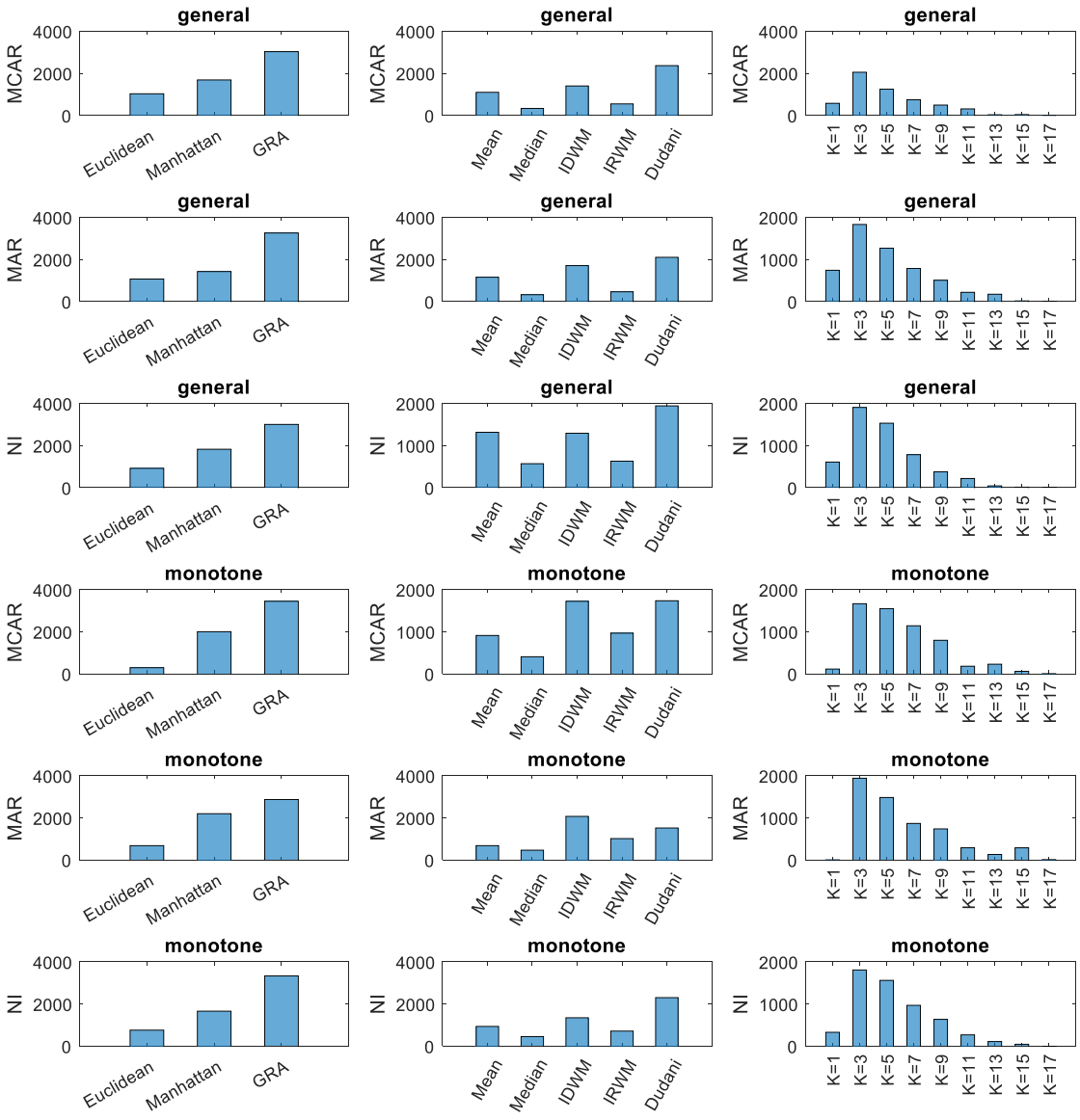

Fig. 10 The parameter distribution of CVB*k*NNI on data ivy at MR = 10%

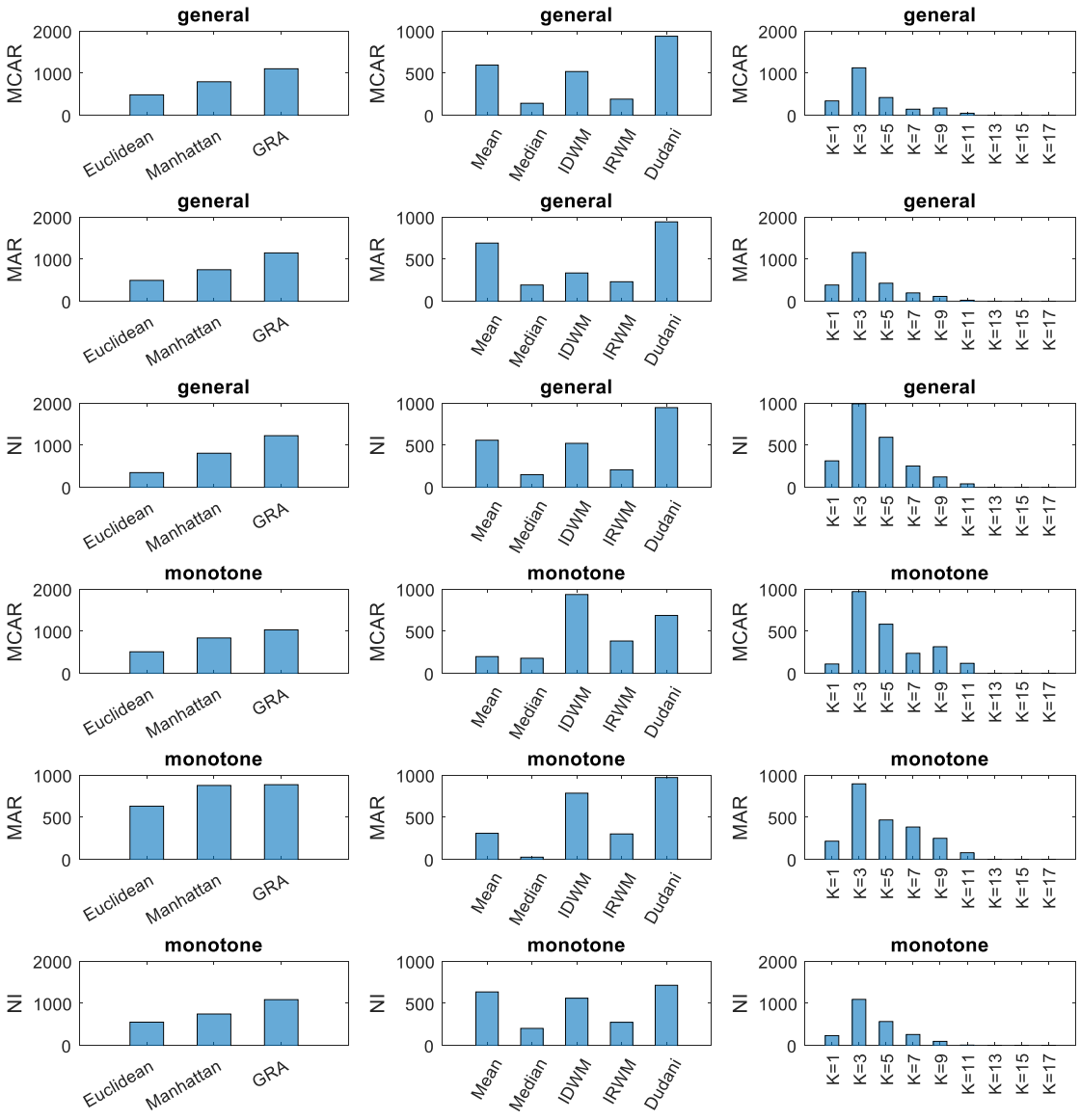

Fig. 11 The parameter distribution of CVB*k*NNI on data KC3 at MR = 10%

Since the estimator of GRA-based distance measure, $K = 3$ with Dudani adaptation method is the most widely selected setting in most cases, this work also evaluates the imputation approach with this selected estimator predefined. This new approach is named as G3D, representing the three predefined parameters. Like Section 5.1, the imputation performance of G3D on each dataset is summarized in terms of various missingness scenarios. The results are shown in Table 14. The

performance is also compared with CVB*k*NNI, FWG*k*NNI, IC*k*NNI, D*k*NNI and MEI. As shown in Table 14, the imputation accuracy of G3D is consistent in general, especially in object-oriented datasets. It is less superior than CVB*k*NNI but superior to FWG*k*NNI, IC*k*NNI, D*k*NNI and MEI in most cases. Although G3D is not the optimal solution by all means, it is recommended being applied in the incomplete software quality datasets as an alternative to the traditional *K*NN imputation strategies. It answers to RQ3.

Left-tail Wilcoxon signed-rank tests are also used to verify if the performance of G3D. For example, in camel data, under the missingness scenario of general MP, MCAR mechanism and 2.5% MR, the average RMSE of 30 replications is 0.098, the '2,3,4,5' in the top-right position denotes that the corresponding 30 RMSE values from G3D imputation are significantly less than that from FWG*k*NNI, IC*k*NNI, D*k*NNI and MEI (See the footnote of Table 14). Therefore, in general, the performance of G3D is better than that of FWG*k*NNI, IC*k*NNI, D*k*NNI and MEI, especially in object-oriented datasets. In arc dataset, some of the average RMSE values are even smaller than that of CVB*k*NNI; however, this relationship is not significant. In the procedural datasets, the size of which is relatively smaller than that of object-oriented datasets. G3D is more frequently not the overall suboptimal imputation. All in all, there is no straightforward evidence that G3D would perform differently under different MMs. It performs worse than IC*k*NNI in procedural datasets if the MP is monotone. But in general, G3D, a *K*NN imputation approach based on MI-based GRA distance measure, $K = 3$ with Dudani adaptation, is better than FWG*k*NNI, IC*k*NNI, D*k*NNI and MEI in most cases in terms of imputation accuracy.

Table 14
The imputation performance of G3D compared with others[1,2]

Data	MP	MR (%)	G3D MCAR	G3D MAR	G3D NI	Data	MP	MR (%)	G3D MCAR	G3D MAR	G3D NI
camel	General	2.5	$0.098^{2,3,4,5}$	$0.091^{2,3,4,5}$	$0.115^{2,3,4,5}$	PC5	General	2.5	$0.065^{2,3,4,5}$	$0.069^{3,4,5}$	$0.086^{2,3,4,5}$
		5	$0.103^{2,3,4,5}$	$0.097^{2,3,4,5}$	$0.124^{2,3,4,5}$			5	$0.069^{2,3,4,5}$	$0.068^{2,3,4,5}$	$0.097^{2,3,4,5}$
		10	$0.111^{2,3,4,5}$	$0.117^{2,3,4,5}$	$0.139^{2,3,4,5}$			10	$0.066^{2,3,4,5}$	$0.070^{2,3,4,5}$	$0.104^{2,3,4,5}$
		20	$0.137^{3,4,5}$	$0.136^{3,4,5}$	$0.171^{2,3,4,5}$			20	$0.072^{2,3,4,5}$	$0.074^{2,3,4,5}$	$0.119^{1,2,3,4,5}$
	Monotone	2.5	$0.110^{2,4,5}$	$0.127^{2,4,5}$	$0.130^{2,3,4,5}$		Monotone	2.5	0.058^{5}	0.061	0.091^{2}
		5	$0.123^{2,3,4,5}$	$0.135^{2,3,4,5}$	$0.132^{2,4,5}$			5	$0.063^{4,5}$	0.070^{5}	$0.090^{2,4,5}$
		10	$0.128^{2,3,4,5}$	$0.131^{2,4,5}$	$0.153^{2,3,4,5}$			10	$0.066^{4,5}$	$0.067^{2,4,5}$	$0.093^{2,4,5}$
		20	$0.132^{2,4,5}$	$0.135^{2,3,4,5}$	$0.176^{2,3,4,5}$			20	$0.067^{2,4,5}$	$0.067^{2,3,4,5}$	$0.118^{1,2,3,4,5}$
ant	General	2.5	$0.104^{2,3,4,5}$	$0.113^{3,4,5}$	$0.123^{2,3,4,5}$	MW1	General	2.5	$0.088^{2,3,4,5}$	$0.094^{2,3,4,5}$	$0.112^{2,3,4,5}$
		5	$0.112^{2,3,4,5}$	$0.116^{2,3,4,5}$	$0.137^{2,3,4,5}$			5	$0.099^{2,3,4,5}$	$0.111^{2,3,4,5}$	$0.130^{2,3,4,5}$
		10	$0.122^{2,3,4,5}$	$0.119^{2,3,4,5}$	$0.150^{2,3,4,5}$			10	$0.111^{2,3,4,5}$	$0.127^{2,3,4,5}$	$0.161^{2,3,4,5}$
		20	$0.139^{3,4,5}$	$0.136^{3,4,5}$	$0.172^{2,3,4,5}$			20	$0.131^{2,3,4,5}$	$0.137^{2,3,4,5}$	$0.196^{2,3,4,5}$
	Monotone	2.5	$0.133^{2,4,5}$	$0.141^{4,5}$	$0.148^{2,3,4,5}$		Monotone	2.5	$0.120^{2,4,5}$	$0.133^{2,4,5}$	0.145^{5}
		5	$0.143^{2,3,4,5}$	$0.151^{4,5}$	$0.147^{2,3,4,5}$			5	$0.127^{4,5}$	$0.140^{2,4,5}$	$0.154^{2,4,5}$
		10	$0.141^{2,3,4,5}$	$0.141^{2,4,5}$	$0.166^{2,3,4,5}$			10	$0.132^{2,4,5}$	$0.137^{2,4,5}$	$0.173^{2,4,5}$

		20	$0.144^{2,4,5}$	$0.143^{2,3,4,5}$	$0.180^{2,3,4,5}$			20	$0.129^{2,4,5}$	$0.139^{2,3,4,5}$	$0.206^{2,4,5}$
ivy	General	2.5	$0.091^{2,3,4,5}$	$0.099^{2,3,4,5}$	$0.114^{2,3,4,5}$	KC3	General	2.5	$0.127^{2,3,4,5}$	$0.110^{2,3,4,5}$	$0.141^{2,3,4,5}$
		5	$0.095^{2,3,4,5}$	$0.099^{2,3,4,5}$	$0.128^{2,3,4,5}$			5	$0.125^{2,3,4,5}$	$0.123^{2,3,4,5}$	$0.167^{2,3,4,5}$
		10	$0.112^{2,3,4,5}$	$0.114^{2,3,4,5}$	$0.151^{2,3,4,5}$			10	$0.125^{2,3,4,5}$	$0.127^{2,3,4,5}$	$0.188^{2,3,4,5}$
		20	$0.140^{3,4,5}$	$0.142^{3,4,5}$	$0.189^{2,3,4,5}$			20	$0.138^{2,3,4,5}$	$0.141^{2,3,4,5}$	$0.220^{2,3,4,5}$
	Monotone	2.5	$0.134^{2,3,4,5}$	$0.126^{2,4,5}$	$0.142^{2,3,4,5}$		Monotone	2.5	0.127^{5}	$0.136^{4,5}$	$0.159^{4,5}$
		5	$0.119^{2,3,4,5}$	$0.132^{2,3,4,5}$	$0.147^{2,3,4,5}$			5	$0.147^{2,4,5}$	$0.144^{2,4,5}$	$0.168^{2,4,5}$
		10	$0.129^{2,3,4,5}$	$0.134^{2,3,4,5}$	$0.160^{2,3,4,5}$			10	$0.146^{2,4,5}$	$0.140^{2,4,5}$	$0.204^{2,5}$
		20	$0.131^{2,4,5}$	$0.136^{2,3,4,5}$	$0.194^{1,2,3,4,5}$			20	$0.147^{2,4,5}$	$0.149^{2,4,5}$	$0.205^{2,5}$
arc	General	2.5	$0.099^{2,3,4,5}$	$0.100^{2,3,4,5}$	$0.116^{2,3,4,5}$	MC2	General	2.5	0.110	0.099^{5}	$0.192^{2,5}$
		5	$0.089^{2,3,4,5}$	$0.094^{2,3,4,5}$	$0.143^{2,3,4,5}$			5	0.132^{3}	0.126	$0.211^{2,3,5}$
		10	$0.106^{2,3,4,5}$	$0.116^{1,2,3,4,5}$	$0.149^{2,3,4,5}$			10	0.151^{3}	0.146^{3}	$0.217^{2,3,4,5}$
		20	$0.146^{2,3,4,5}$	$0.147^{2,3,4,5}$	$0.212^{2,3,4,5}$			20	$0.155^{3,5}$	$0.155^{3,5}$	$0.237^{1,3,4,5}$
	Monotone	2.5	$0.104^{1,2,3,4,5}$	$0.130^{2,3,4,5}$	$0.122^{2,3,4,5}$		Monotone	2.5	0.160^{5}	0.123^{3}	$0.185^{2,5}$
		5	$0.129^{2,3,4,5}$	$0.131^{2,3,4,5}$	$0.140^{2,3,4,5}$			5	0.138^{5}	0.136^{5}	0.202^{2}
		10	$0.135^{2,3,4,5}$	$0.130^{2,3,4,5}$	$0.174^{2,3,4,5}$			10	$0.125^{4,5}$	0.131^{5}	0.200^{2}
		20	$0.136^{2,3,4,5}$	$0.144^{2,3,4,5}$	$0.215^{2,3,4,5}$			20	$0.146^{2,5}$	$0.134^{2,4,5}$	0.231^{2}

[1]Filled in green: G3D performs better than CVB*k*NNI; Filled in blue: G3D performs better than FWG*k*NNI (Compare with Table 8).
[2]1 = CVB*k*NNI, 2 = FWG*k*NNI, 3 = IC*k*NNI, 4 = D*k*NNI, 5 = MEI

5.4 Classification accuracy

This part assesses the imputation effectiveness from another perspective, which compares different ML classifiers (*K*NN, Discriminant analysis, Naive Bayes and SVM) built on the complete data constructed after the imputation. It answers to RQ4 to compare and verify CA. Since the classification models cannot be exhaustively applied, most studies regarded this procedure as an auxiliary step evaluating the imputation performance.

To present the impact of imputation on fault-proneness classification accuracy, four ML classifiers on the two relatively large imputed datasets: PC5 and camel, are conducted. To save space, only the data versions with MR = 10% are analyzed for comparison. The other missingness scenarios are kept as well. For each classifier, CA is computed after a 10-fold cross-validation. G3D is also included in the experiment.

Table 15 presents the results of CA. Note that 'No imputation' means we use the original data (w/o missingness simulation) for classification. 'No imputation', therefore, is a benchmark in the comparison. The imputed data is less distorted if the classification performance on which is as usual as that on the corresponding original one. The results clearly show that the classifiers based on either the original dataset or the incomplete one with MEI used generally present relatively worse classification performance. In terms of the CA after imputation, CVB*k*NNI and G3D are generally superior to others in most cases. And the IC*k*NNI has a suboptimal performance. In

terms of the classifiers, *K*NN, Discriminant analysis, and Naïve Bayes are more sensitive to imputation approaches than SVM. The performance of SVM is even not sensitive to MM or MP. In general, the results show that appropriate imputation approach could be beneficial to the CA of specific classifier. To sum, when using CVB*k*NNI and G3D as the imputation approach on the incomplete data, the classification bias could also be maintained or even reduced in commonly used classification tasks.

Table 15
The comparison of CA using different ML classifiers on data camel and PC5 with MR = 10%

Data	MM	Imputation Approaches	Classifiers and MP							
			General				Monotone			
			KNN (K = 5)	Discriminant Analysis	Naive Bayes	SVM	KNN (K = 5)	Discriminant Analysis	Naive Bayes	SVM
camel	MCAR	No imputation	0.770[1]	0.785	0.775	0.792	0.776	0.786	0.770	0.792
		CVB*k*NNI	0.794	0.798	0.776	0.794	0.795	0.796	0.774	0.794
		G3D	0.793	0.792	0.768	0.793	0.792	0.794	0.771	0.793
		FWG*k*NNI	0.773	0.791	0.776	0.792	0.790	0.785	0.769	0.793
		IC*k*NNI	0.780	0.795	0.777	0.794	0.789	0.794	0.773	0.793
		D*k*NNI	0.779	0.793	0.771	0.793	0.779	0.795	0.769	0.793
		MEI	0.771	0.792	0.771	0.793	0.770	0.791	0.770	0.791
	MAR	No imputation	0.770	0.786	0.771	0.792	0.776	0.783	0.771	0.793
		CVB*k*NNI	0.794	0.797	0.772	0.794	0.795	0.796	0.777	0.794
		G3D	0.794	0.793	0.772	0.794	0.795	0.795	0.768	0.794
		FWG*k*NNI	0.784	0.792	0.770	0.793	0.790	0.785	0.770	0.793
		IC*k*NNI	0.785	0.797	0.783	0.797	0.789	0.788	0.773	0.793
		D*k*NNI	0.787	0.792	0.770	0.794	0.779	0.792	0.768	0.793
		MEI	0.782	0.785	0.769	0.792	0.770	0.784	0.771	0.792
	NI	No imputation	0.776	0.783	0.775	0.793	0.774	0.791	0.773	0.792
		CVB*k*NNI	0.795	0.794	0.775	0.794	0.792	0.796	0.774	0.794
		G3D	0.794	0.798	0.774	0.794	0.794	0.797	0.771	0.793
		FWG*k*NNI	0.785	0.789	0.775	0.794	0.783	0.792	0.770	0.793
		IC*k*NNI	0.789	0.793	0.780	0.795	0.784	0.792	0.771	0.793
		D*k*NNI	0.780	0.794	0.772	0.792	0.783	0.790	0.765	0.794
		MEI	0.776	0.788	0.768	0.792	0.772	0.786	0.768	0.794
PC5	MCAR	No imputation	0.768	0.776	0.705	0.779	0.771	0.774	0.705	0.779
		CVB*k*NNI	0.781	0.783	0.709	0.780	0.790	0.781	0.708	0.780
		G3D	0.778	0.779	0.707	0.780	0.782	0.782	0.712	0.781
		FWG*k*NNI	0.772	0.772	0.706	0.780	0.781	0.775	0.702	0.780
		IC*k*NNI	0.787	0.783	0.713	0.781	0.782	0.779	0.706	0.780
		D*k*NNI	0.785	0.776	0.702	0.780	0.779	0.777	0.703	0.780
		MEI	0.766	0.774	0.709	0.780	0.760	0.779	0.706	0.780
	MAR	No imputation	0.775	0.768	0.699	0.780	0.775	0.776	0.701	0.779
		CVB*k*NNI	0.776	0.786	0.704	0.781	0.786	0.789	0.707	0.780
		G3D	0.785	0.779	0.705	0.781	0.777	0.776	0.710	0.781
		FWG*k*NNI	0.765	0.773	0.699	0.779	0.782	0.776	0.706	0.779
		IC*k*NNI	0.786	0.786	0.704	0.780	0.776	0.775	0.710	0.779
		D*k*NNI	0.785	0.778	0.700	0.779	0.773	0.781	0.698	0.779
		MEI	0.763	0.773	0.700	0.779	0.758	0.778	0.703	0.779
	NI	No imputation	0.776	0.775	0.706	0.779	0.768	0.774	0.694	0.779
		CVB*k*NNI	0.783	0.780	0.711	0.781	0.776	0.788	0.700	0.781
		G3D	0.779	0.786	0.704	0.780	0.772	0.777	0.697	0.779
		FWG*k*NNI	0.759	0.775	0.709	0.780	0.768	0.777	0.695	0.780
		IC*k*NNI	0.787	0.784	0.717	0.784	0.776	0.776	0.700	0.779
		D*k*NNI	0.773	0.777	0.708	0.780	0.774	0.775	0.698	0.780
		MEI	0.756	0.775	0.703	0.780	0.754	0.772	0.693	0.779

[1]The maximum CAs in each condition are in green; the minimum CAs are in red; while the 2nd largest CAs are marked in blue.

6. Threats to validity

The threats to validity are generally distributed into four groups: conclusion, internal, construct, and external validity. The conclusion validity is related to the ability to draw significant correct conclusions; regarding which, we carefully applied the statistical tests, showing statistical significance for the obtained results. Moreover, we have used two relatively large datasets (camel and PC5) to mitigate the threats related to the number of observations composing the datasets.

The construct validity refers to the agreement between a theoretical concept and a specific measure. As to the evaluation of different *K* nearest neighbor (*K*NN) imputation approaches, we made use of one balanced performance measure and 8 public software quality datasets. The data repository used in the work has been previously used in numerous empirical quality studies.

As the study concentrates on a structural investigation of a novel *K*NN imputation approach, the internal validity on experiment design is presented in one aspect: CVB*k*NNI is a computation-consuming way to improve imputation accuracy. However, if the imputation accuracy is the top priority, the value of CVB*k*NNI is then obvious. The proposal of G3D stands as an alternative to mitigate this issue as well.

The threats to external validity are controlled well in this study. Eight object-oriented and procedural software quality datasets are examined in the study. Meanwhile, 3 missingness mechanisms (MMs), 2 missingness patterns (MPs), and 4 missingness ratios (MRs) are also considered during missingness simulation for testing the performance of the proposed imputation approach under various conditions. One issue is that we did not consider adopting more different kinds of imputation approaches, for example, the Bayes multiple imputation (BMI), as the competitor to CVB*k*NNI. This study only focuses on the improvement and comparison of *K*NN related imputation approaches. In the future work, more comparison studies shall be explored.

7. Conclusions and future work

This empirical study proposes a novel *K* Nearest Neighbor (*K*NN) based imputation approach: called CVB*k*NNI, and its improved performance has been validated in the software quality prediction domain. CVB*k*NNI is different to other approaches since it does not have predetermined fixed estimator, and instead it adaptively selects the optimal estimator for each missing value in the dataset. The estimator of CVB*k*NNI includes a pool of three distance measures, multiple choices of *K* values together with five adaptation methods. Our result shows that CVB*k*NNI outperforms other competing approaches in overall imputation accuracy. From the returned estimator of the CVB*k*NNI, the optimal parameter combination of *K*NN imputation for software quality dataset is then correctly determined, which is named as G3D. Further evaluations on the CVB*k*NNI have been performed, specifically on incomplete datasets and compared several other competing approaches.

In particular, 4 findings are noteworthy from the study:

(1) Our proposed cross-validation based *K*NN imputation could further improve the imputation performance on software quality datasets, in which calculating the feature relevance during measuring the pair distance is very necessary.

(2) The impact of missingness mechanisms and patterns on imputation performance exists. Non-ignorable missingness mechanism could significantly impact the imputation accuracy. The impact of missingness pattern is related to the dataset.

(3) The performance of fault-proneness classification is acceptable when CVB*k*NNI was used as the preprocessing method.

(4) For *K*NN imputation using $K = 3$ and Dudani adaptation, together with the distance measure based on mutual information weighted grey relational analysis, is considered ideal and recommended for incomplete software quality datasets.

CVB*k*NNI could be easily applicable to other domains in software engineering, which are subject to further investigations in our future work. Theoretically it would further improve imputation accuracy when dealing with incomplete datasets, but also helps to find the optimal *K*NN imputation algorithm under different circumstance. Besides, determining more meaningful parameter configurations or components to further improve the accuracy of CVB*k*NNI is also being investigated.

Acknowledgement

The authors would like to thank the anonymous reviewers for their invaluable feedback. This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 125113, 11200015 and 11214116), and the research funds of City University of Hong Kong (No. 7004683 and 7004474).

References

Arcuri, A., Briand, L., 2014. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Journal of Software: Testing, Verification and Reliability 24, 219-250.
Azzeh, M., 2012. A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation. Empirical Softw Eng 17, 90-127.
Bansiya, J., Davis, C.G., 2002. A hierarchical model for object-oriented design quality assessment. IEEE Trans Softw Eng 28, 4-17.
Brás, L.P., Menezes, J.C., 2007. Improving cluster-based missing value estimation of DNA microarray data. Biomol Eng 24, 273-282.
Cartwright, M., Shepperd, M.J., Song, Q., 2003. Dealing with missing software project data, 9th International Software Metrics Symposium (METRIC'03) Sydney, Australia, pp. 154-165.
Caruana, R., 2001. A non-parametric EM-style algorithm for imputing missing values, the 4th International Conference of Artificial Intelligence and Statistics (AISTATS'01), Key West, Florida, USA.
Chen, J., Shao, J., 2000. Nearest neighbor imputation for survey data. Journal of Official Statistics 16, 113-132.
Chidamber, S.R., Kemerer, C.F., 1994. A metrics suite for object oriented design. IEEE Trans Softw Eng 20, 476-493.
Conversano, C., Siciliano, R., 2009. Incremental tree-based missing data imputation with lexicographic ordering. J Classif 26, 361-379.
Deb, R., Liew, A.W.-C., 2016. Missing value imputation for the analysis of incomplete traffic accident data. Inform Sciences 339, 274-289.
Duda, R.O., Hart, P.E., 1973. Pattern classification and scene analysis. Wiley New York.
Dudani, S.A., 1976. The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man and Cybernetics, 325-327.
Finley, A.O., McRoberts, R.E., Ek, A.R., 2006. Applying an efficient k-nearest neighbor search to forest attribute imputation. Forest Sci 52, 130-135.
Foss, T., Stensrud, E., Kitchenham, B., Myrtveit, I., 2003. A simulation study of the model evaluation criterion MMRE. IEEE Trans Softw Eng 29, 985-995.
García-Laencina, P.J., Sancho-Gómez, J.-L., Figueiras-Vidal, A.R., Verleysen, M., 2009. K nearest neighbours with mutual information for simultaneous classification and missing data imputation. Neurocomputing 72, 1483-1493.
Halstead, M.H., 1977. Elements of software science (Operating and programming systems series). Elsevier Science Inc.
Henderson-Sellers, B., 1996. Object-oriented metrics: Measures of complexity. Prentice-Hall, Inc.

Hron, K., Templ, M., Filzmoser, P., 2010. Imputation of missing values for compositional data using classical and robust methods. Comput Stat Data An 54, 3095-3107.
Huang, C.-C., Lee, H.-M., 2004. A grey-based nearest neighbor approach for missing attribute value prediction. Appl Intell 20, 239-252.
Huang, J., Li, Y.F., Xie, M., 2015. An empirical analysis of data preprocessing for machine learning-based software cost estimation. Inf Softw Technol 67, 108-127.
Jing, X.-Y., Qi, F., Wu, F., Xu, B., 2016. Missing data imputation based on low-rank recovery and semi-supervised regression for software effort estimation, the 38th International Conference on Software Engineering (ICSE'16). ACM, Austin, TX, USA, pp. 607-618.
Jönsson, P., Wohlin, C., 2006. Benchmarking k-nearest neighbour imputation with homogeneous Likert data. Empirical Softw Eng 11, 463-489.
Jureczko, M., Madeyski, L., 2010. Towards identifying software project clusters with regard to defect prediction, the 6th International Conference on Predictive Models in Software Engineering (PROMISE'10). ACM, Timisoara, Romania, p. 9.
Khatibi Bardsiri, V., Jawawi, D.N.A., Hashim, S.Z.M., Khatibi, E., 2013. A PSO-based model to increase the accuracy of software development effort estimation. Softw Qual J 21, 501-526.
Khoshgoftaar, T.M., Van Hulse, J., 2008. Imputation techniques for multivariate missingness in software measurement data. Softw Qual J 16, 563-600.
Kocaguneli, E., Member, S., Menzies, T., 2012a. Exploiting the essential assumptions of analogy-based effort estimation. IEEE Trans Softw Eng 38, 425-439.
Kocaguneli, E., Member, S., Menzies, T., 2012b. On the value of ensemble effort estimation. IEEE Trans Softw Eng 38, 1403-1416.
Kocaguneli, E., Menzies, T., Keung, J., Cok, D., Madachy, R., 2013a. Active learning and effort estimation: Finding the essential content of software effort estimation data. IEEE Trans Softw Eng 39, 1040-1053.
Kocaguneli, E., Menzies, T., Keung, J.W., 2013b. Kernel methods for software effort estimation: Effects of different kernel functions and bandwidths on estimation accuracy. Empirical Softw Eng 18, 1-24.
Kullback, S., 1997. Information theory and statistics. Courier Corporation.
Lall, U., Sharma, A., 1996. A nearest neighbor bootstrap for resampling hydrologic time series. Water Resour Res 32, 679-693.
Li, D., Deogun, J., Spaulding, W., Shuart, B., 2004. Towards missing data imputation: A study of fuzzy k-means clustering method, the 4th International Conference of Rough Sets and Current Trends in Computing (RSCTC'04), Uppsala, Sweden, pp. 573-579.
Li, J., Al-Emran, A., Ruhe, G., 2007. Impact analysis of missing values on the prediction accuracy of analogy-based software effort estimation method AQUA, the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM'07), Madrid, Spain, pp. 126-135.
Li, Y.F., Xie, M., Goh, T.N., 2009a. A study of mutual information based feature selection for case based reasoning in software cost estimation. Expert Syst Appl 36, 5921-5931.
Li, Y.F., Xie, M., Goh, T.N., 2009b. A study of project selection and feature weighting for analogy based software cost estimation. J Syst Softw 82, 241-252.
Little, R.J., Rubin, D.B., 2002. Statistical analysis with missing data. John Wiley & Sons, Inc.
Ma, X., Zhong, Q., 2016. Missing value imputation method for disaster decision-making using K nearest neighbor. J Appl Stat 43, 767-781.

Magnussen, S., Tomppo, E., 2014. The k-nearest neighbor technique with local linear regression. Scand J Forest Res 29, 120-131.
Maier, M., Hein, M., von Luxburg, U., 2009. Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. Theor Comput Sci 410, 1749-1764.
Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M., Webster, S., 2000. Investigation of machine learning based prediction systems. J Syst Softw 53, 23-29.
Martin, R., 1994. OO design quality metrics - An analysis of dependencies, the 9th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'94), Portland, Oregon, USA, pp. 151-170.
McCabe, T.J., 1976. A complexity measure. IEEE Trans Softw Eng 2, 308-320.
Mendes, E., Watson, I., Triggs, C., Mosley, N., Counsell, S., 2003. A comparative study of cost estimation models for web hypermedia applications. Empirical Softw Eng 8, 163-196.
Menzies, T., Krishna, R., Pryor, D., 2016. The PROMISE Repository of empirical software engineering data. North Carolina State University, Department of Computer Science, http://openscience.us/repo.
Minku, L.L., Yao, X., 2011. A principled evaluation of ensembles of learning machines for software effort estimation, the 7th International Conference on Predictive Models in Software Engineering (PROMISE'11), Banff, Canada, pp. 1-10.
Mittas, N., Angelis, L., 2010. LSEbA: least squares regression and estimation by analogy in a semi-parametric model for software cost estimation. Empirical Softw Eng 15, 523-555.
Mockus, A., 2008. Missing data in software engineering, Guide to advanced empirical software engineering. Springer, pp. 185-200.
Myrtveit, I., Stensrud, E., Olsson, U.H., 2001. Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods. IEEE Trans Softw Eng 27, 999-1013.
Pan, R., Yang, T., Cao, J., Lu, K., Zhang, Z., 2015. Missing data imputation by K nearest neighbours based on grey relational structure and mutual information. Appl Intell, 1-19.
Peng, H., Long, F., Ding, C., 2005. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell 27, 1226-1238.
Poloczek, J., Treiber, N.A., Kramer, O., 2014. KNN regression as geo-imputation method for spatio-temporal wind data, International Joint Conference SOCO'14-CISIS'14-ICEUTE'14. Springer, pp. 185-193.
Rey-del-Castillo, P., Cardeñosa, J., 2012. Fuzzy min–max neural networks for categorical data: application to missing data imputation. Neural Computing and Applications 21, 1349-1362.
Sahri, Z., Yusof, R., Watada, J., 2014. FINNIM: Iterative imputation of missing values in dissolved gas analysis dataset. IEEE Transactions on Industrial Informatics 10, 2093-2102.
Sarro, F., Petrozziello, A., Harman, M., 2016. Multi-objective software effort estimation, the 38th International Conference on Software Engineering (ICSE'16). ACM, Austin, TX, USA, pp. 619-630.
Sentas, P., Angelis, L., 2006. Categorical missing data imputation for software cost estimation by multinomial logistic regression. J Syst Softw 79, 404-414.
Shepperd, M., Schofield, C., 1997. Estimating software project effort using analogies. IEEE Trans Softw Eng 23, 736-743.
Shepperd, M., Song, Q., Sun, Z., Mair, C., 2013. Data quality: Some comments on the nasa software defect datasets. IEEE Trans Softw Eng 39, 1208-1215.

Silva-Ramírez, E.-L., Pino-Mejías, R., López-Coello, M., 2015. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. Applied Soft Computing 29, 65-74.
Song, Q., Shepperd, M., 2007. A new imputation method for small software project data sets. J Syst Softw 80, 51-62.
Song, Q., Shepperd, M., Cartwright, M., 2005. A short note on safest default missingness mechanism assumptions. Empirical Softw Eng 10, 235-243.
Song, Q., Shepperd, M., Chen, X., Liu, J., 2008. Can k-NN imputation improve the performance of C4. 5 with small software project data sets? A comparative evaluation. J Syst Softw 81, 2361-2370.
Stekhoven, D.J., Bühlmann, P., 2012. MissForest - Non-parametric missing value imputation for mixed-type data. Bioinformatics 28, 112-118.
Strike, K., Emam, K.E., Madhavji, N., 2001. Software cost estimation with incomplete data. IEEE Trans Softw Eng 27, 890-908.
Suyundikov, A., Stevens, J.R., Corcoran, C., Herrick, J., Wolff, R.K., Slattery, M.L., 2015. Accounting for dependence induced by weighted KNN imputation in paired samples, motivated by a colorectal cancer study. PLoS ONE 10, e0119876.
Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B., 2001. Missing value estimation methods for DNA microarrays. Bioinformatics 17, 520-525.
Twala, B., Cartwright, M., Shepperd, M., 2005. Comparison of various methods for handling incomplete data in software engineering databases, the 2005 International Symposium on Empirical Software Engineering (ISESE'05), Noosa Heads, Australia, pp. 105-114.
Valdiviezo, H.C., Van Aelst, S., 2015. Tree-based prediction on incomplete data using imputation or surrogate decisions. Inform Sciences 311, 163-181.
Van Buuren, S., 2012. Flexible imputation of missing data. CRC press.
Van Hulse, J., Khoshgoftaar, T.M., 2014. Incomplete-case nearest neighbor imputation in software measurement data. Inform Sciences 259, 596-610.
Walkerden, F., Jeffery, R., 1999. Empirical study of analogy-based software effort estimation. Empirical Softw Eng 4, 135-158.
Zhang, S., 2011. Shell-neighbor method and its application in missing data imputation. Appl Intell 35, 123-133.
Zhang, S., 2012. Nearest neighbor selection for iteratively kNN imputation. J Syst Softw 85, 2541-2552.
Zhang, S., Jin, Z., Zhu, X., 2011. Missing data imputation by utilizing information within incomplete instances. J Syst Softw 84, 452-459.
Zhang, S., Li, X., Zong, M., Zhu, X., Cheng, D., 2017. Learning k for kNN Classification. ACM Trans. Intell. Syst. Technol. 8, 1-19.