Received: February 7, 2021. Revised: March 8, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Missing Data Imputation Via Stacked Denoising Autoencoder Combined with Dropout Regularization Based Small Dataset in Software Effort Estimation

Robert Marco[1]* **Sharifah Sakinah Syed Ahmad[2]** **Sabrina Ahmad[2]**

[1]*Department of Informatics, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia*
[2]*Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia*
* Corresponding author's Email: robertmarco@amikom.ac.id

Abstract: Missing data in software attributes as a common occurrence that can lead to incorrect estimates and poor predictability. For this reason, the imputation approach was used in our study. We model incomplete data using an Autoencoder (AE) approach for missing value imputation, which substantially minimizes the complexity of data modeling when dealing with small data in regression situations. In this study, we introduce the imputation approach through Stacked Denoising Autoencoder (SDAE) with dropout regularization and perform hyperparameter tuning. The scenario presented in this research is built up by eliminating data intervals from the whole dataset at various missing rates (10 percent - 80 percent missing rate). Experimental findings from 11 datasets support the efficacy of the suggested technique (on Promise and ISBSG10). The proposed method's performance is compared to that of other techniques (such as: Generative Adversarial Imputation Networks (GAIN), k-Nearest Neighbor imputation (kNNI), Multiple Imputation by Chained Equations (MICE), and Random Forest Imputation (MissForest)), demonstrating that our method outperforms others even with an 80% data missing rate, with producing the best value (low) on mean absolute error (MAE: 0.0839) and root mean square error (RMSE: 0.1061).

Keywords: Stacked, Denoising autoencoder, Dropout regularization, Inducing missingness, Software effort estimation.

1. Introduction

Missing data in critical software properties is widespread, and it can lead to inaccurate estimations and poor predictability [1], [2], It can sabotage the learning process by leading to incorrect assumptions [3]. Furthermore, it can result in data analysis bias and loss of information [4]. Understanding the mechanics of missing data is necessary for comprehending the impact of missing data on a particular analysis or method of missing data [5], [6]. There are three processes explaining the pattern of missing values, according to little and rubin (1987), such as: missing at random (MAR), missing completely at random (MCAR), and nonignorable missing (NIM) [6].

Meanwhile, in the missing data methodology, there are three ways to deal with missing data [7], such as: toleration, deletion technique, and imputation technique. Although simple, toleration is not a reliable approximation, and sometimes even provides a less efficient estimate than the estimation of the deletion technique [5], [6]. Meanwhile, the deletion technique has many disadvantages after deletion of valuable data including a loss of precision and result bias [8]. Furthermore, numerous studies have revealed the risks of utilizing listwise deletion [9].

The missing data imputation methodology links the missing data before applying the standard full data method to the filled data [8]. Popular static imputation method packages in the SEE field accessible in R, including as multiple imputations by chained equation (MICE), random forest imputation (MissForest), and k-nearest neighbor imputation (kNNI) [10]. Some of these imputation approach procedures have previously been studied in empirical software engineering in the realm of software

measurement data.

The application of the kNNI approach to a model with insufficient capacity for the task. MICE and MissForest, on the other hand, are iterative approaches that try to enter the whole data set all at once. In real-world scenarios, when new data is generated on a daily basis, this can be a disadvantage [11]. According to Idri et al (2015), Despite the fact that most software project data sets contain largely categorical characteristics with a large number of missing values, most imputation methods are used on numeric data types [12].

However, some of these imputation methods cannot handle missing data with multi-type variables, for example, binary, categorical, and continuous attribute combinations, and do not have outliers resistance. Meanwhile, various deep learning-based methods have been developed in recent years to overcome this challenge, such as: generative adversarial imputation networks (GAIN) [13, 14] and denoising autoencoders (DAE) [15–17]. However, because the GAIN technique assumes the data is static and ignores the temporal component of the data, it typically performs badly when the data is a long run time series [16], and network modeling is tough to do, as well as training [15]. In contrast to DAE, which is designed to recover missing data from noisy input through unattended learning, which makes it suitable for unlabeled data [18]. However, missing data may depend on non-observable latent interactions or representations in the input data set space [15]. DAE can use stochastic noise injection to corrupt a subset of input data and then use nested nonlinear transformations to try to reconstruct it [19], and if you work on a known small training data set, there will be an overfitting problem [20]. Unfortunately, this kind of algorithm usually faces a series of drawbacks, such as low training efficiency, complicated network, local minimums, difficult tuning of control parameters, and gradient disappearing [21].

Nevertheless, according to Vincent et al (2008), the denoising autoencoder (DAE) was created to remove noise from data with high dimensions in hidden layers and stochastic contamination in inputs [22]. The data distribution is implicitly approximated by the DAE reconstruction capabilities as an asymptotic markov chain distribution alternating between corruption and denoising [23]. In the meantime, the stacked method is being used to limit process variability and corruption's influence [24]. As a result, the stacked denoising autoencoder (SDAE) can handle data sets that are more complex (higher number of samples and dimensions). Furthermore, while the benefits of SDAE appear to be greater at larger loss rates (by 40 percent) [25]. To limit the possibility of overfitting, we used a dropout strategy to train this newly rebuilt DAE, which extends the corruption process deeper into the neural network architecture [19].

We present a stacked DAE-based model of missing data imputation adapted for missing imputation in regression models with small data sets, motivated by the observed drawbacks of implementing a combined DAE utilizing stacked on the missing imputation problem. Thus, our research, which employs a stacked DAE (SDAE) based on dropout regularization approaches, intends to uncover hidden correlations between missing and non-missing values in small data sets, and then estimate for imputations [26], [20].

The rest of this paper is laid out as follows. In section 2, we will review related studies, and in section 3, we will present our approach to using a stacked denoising autoencoder and the method we propose. In our papers in section 4, will discuss research methodology. Section 5 presents performance evaluation and compares our proposed technique to others. Section 6 concludes with a discussion of the conclusion and future work.

2. Related studies

Since we propose a new data imputation method for the problem of missing data on a small dataset on a regression problem, we review previous work on imputation, for example, autoencoder denoising which has received almost no attention in software engineering.

Yoon et al. (2018) used five UCI machine learning repository datasets to evaluate GAIN performance. The experimental results show that GAIN has the best RMSE value on the breast dataset (0.0546), spam (0.0513), letter (0.1198), credit (0.1858), and news (0.1441). while the overall MSE value obtained a value of 0.1137 [13]. Meanwhile, Stekhoven et al. (2012) evaluated missforest using ten datasets. MissForest performs better than MICE and kNNI, again reducing imputation errors in most cases by >50 % [27]. Van Buuren and Oudshoorn (2011), documenting a significant MICE update, results show that imputation using the fully conditional specification (FCS) will prove to be a great addition to our statistical tool [28]. A study conducted by [13], showed that MICE tested using the UCI machine learning repository obtained the best RMSE scores on the breast dataset (0.0646), spam (0.0699), letter (0.1537), credit (0.2585), and news (0.1763). While the overall MSE value obtained a value of 0.9467. Meanwhile, a study conducted by Abnane and Idri (2018) evaluated the missing data

imputation technique using four datasets (ISBSG, cocomo81, USP05FT, and USP05RQ). The results show that the standard value of accuracy using the MCAR mechanism, the kNNI technique by 41 %, outperforms the tolerance and deletion technique. While in the MAR mechanism, kNNI is 34 % and the NIM mechanism shows kNNI is 32 % [29].

Huamin et al. (2020) proposed a denoising autoencoder (DAE) based on time series data representation by reconstructing the data using a recurrence plot (RP) and a Gramian angular field (GAF). The experimental results using MSE gave values to the data of ECG200 (GAF: 0.0048; RP: 0.0037), face all (GAF: 0.0134; RP: 0.0221), swedish leaf (GAF: 0.0098; RP: 0.0092), OSU leaf (GAF: 0.0077; RP:0.0121), wafer (GAF:0.0240; RP:0.0069), 50 words (GAF:0.0101; RP:0.0108), and coffee (GAF:0.0336; RP:0.0234). However, this method is only effective for univariate time series, but not for more complicated multivariate [15]. Zhang and Yin (2019) proposed imputing missing data from multivariate time series by adapting long short term-memory (LSTM) and denoising autoencoder (DAE). This study evaluates clinical vital signs dataset using RMSE with accuracy values on gesture (0.175), eye state (0.260), occupancy (0.318), EEG (0.060), and eICU (0.046) data. This method can reduce the RMSE value by 70 % [16]. Zhang and Yin (2019) proposed using long short term memory (LSTM) and denoising autoencoder (DAE) to impute missing data from multivariate time series. The accuracy of gesture (0.175), eye state (0.260), occupancy (0.318), EEG (0.060), and eICU (0.046) data was evaluated using RMSE. The RMSE number can be reduced by 70 % with this strategy. When a data collection with cryptic classes is utilized to create multi-modal data, however, the results can be quite different [17]. Meanwhile, Tihon et al. (2021) published DAE with mask attention (DAEMA) in the UCI repository. This method outperforms current approaches on multiple samples of missing data under MCAR and MNAR. Unfortunately, this method shows poor performance when dealing with small datasets, so DAEMA requires a sufficient amount of data to model the distribution of the data [11]. Gondara and Wang (2018) proposed a multiple imputation model based on denoising autoencoders (MIDA). Our model outperforms the competition by achieving lower RMSE values even with small sample numbers, which is a difficult issue for deep architectures [3]. Meanwhile, Lu et al. (2020) propose a multiple imputation model based on the denoising autoencoder to investigate the internal representation of data using the metamorphic truth and imputation feedback mechanisms to maintain statistical integrity of attributes and eliminate bias in the learning process. Our model outperforms the DAE and MICE imputation models. Nonetheless, when it comes to covariance deviations, it's evident that the bias introduced by the original imputation has a large impact on covariance [30].

Based on the results of the literature review that we have done. We reviewed several other popular imputation methods in the machine learning field. Unfortunately, these methods are not noise and outlier-resistant. Next, we also examine several imputation methods based on the denoising autoencoder by outlining some of the limitations of previous studies. Motivated by these shortcomings, we propose an enhanced stacked DAE with dropout regularization for regression problems on small data sets and multi-variable types. In contrast to several DAE methods that have been developed previously, our method adds a standardization technique that aims to speed up computational time. Because we use a small dataset, we add a dropout regularization technique in order to reduce the computational cost of overfitting the model. Stacked tries to limit the diversity of processes and the effects of corruption. This is the first large study based our knowledge that uses deep learning to evaluate the imputation of missing data in the context of software effort estimation.

3. Our approach

3.1 Autoencoder

The autoencoder (AE) is an unsupervised learning approach that use a neural network to learn encoding or efficient data representation in order to reconstruct the original input data [31], [32]. The autoencoder consists of two parts: an encoder and a decoder, each denoted by the letters f and g. The construction of the autoencoder is shown in Fig. 1.

The mapping function in Eq. (1) is used by the encoder to transfer the input vector $x \in \mathbb{R}^n$ to a hidden representation $y \in \mathbb{R}^m$ [31], [32].

$$y = f_\theta(x) = s(Wx + b) \tag{1}$$

Where, $\theta = \{W, b\}$, W is $m \times n$ for the weight of the matrix, The bias vector is b and the activation function is s (eg sigmoid or rectified linear unit). The decoder retrieves the hidden representation y to map it to the reconstructed vector $z \in \mathbb{R}^n$ using Eq. (2).

$$z = \mathrm{g}_\theta{}'(y) = s'(W'y + b') \tag{2}$$

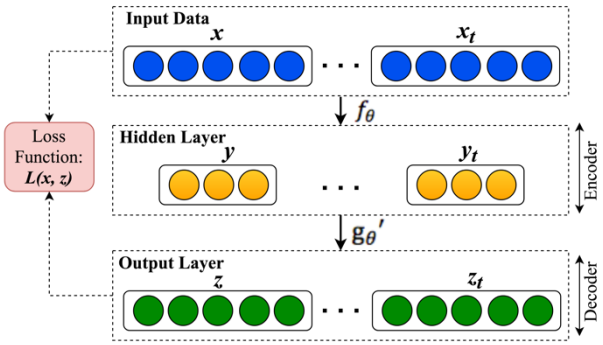

Figure. 1 Architecture standard autoencoder

Where, $\theta' = \{W', b'\}$, W' is $m \times n$ for the weight of the matrix, while b' is vector bias, and s′ is the activation function. As a result, the autoencoder's parameter θ and θ' will be tuned to minimize the average reconstruction error, using Eq (3).

$$\theta^*, \theta' * = argmin \frac{1}{n} \sum_{i=1}^{n} L\left(x^{(i)}, z^{(i)}\right) = argmin \frac{1}{n} \sum_{i=1}^{n} L(x^{(i)}, g_\theta'(f_\theta(x^{(i)}))) \quad (3)$$

Where, $\theta^*, \theta' *$ is a parameter that must be learned on the data. Thus, the autoencoder can minimize reconstruction losses between input data and output data, such as MSE loss. The autoencoder's goal is to reconstruct the $z^{(i)}$ in such a way that $z^{(i)} \approx x^{(i)}$ is reconstructed with the least amount of function loss $L\left(x^{(i)}, z^{(i)}\right)$. For continuous data, this function is defined as the mean squared error, while for discrete data, it is defined as the cross entropy. It is used in our imputation to reduce the squared error between the input x and the output z which is reconstructed using Eq. (4), for the real value x.

$$L\left(x^{(i)}, z^{(i)}\right) = C(\sigma^2)||x - z||^2 \quad (4)$$

Where, $C(\sigma^2)$ specifies a constant that is solely dependent on σ^2 and can be discarded for optimization purposes. The squared error inherent in most traditional autoencoders is designed to achieve this goal. Because of the Gaussian interpretation, it is more natural not to utilize nonlinearity squashing in the decoder in this situation [33].

3.2 Denoising autoencoder

The denoising autoencoder, as shown in Fig. 2, functions similarly to the encoder but adds noise to the input data. Denoising autoencoder development to reconstruct input x for corrupted $\tilde{x}$ version, which is a more difficult work than the basic autoencoder.

This is accomplished by first corrupting the initial

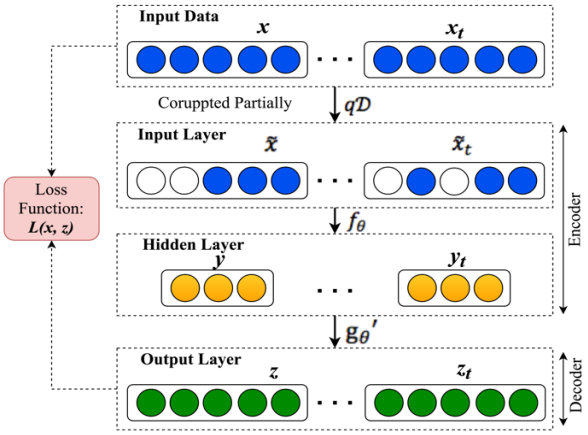

Figure. 2 Architecture denoising autoencoder

input x into $\tilde{x}$ via $\tilde{x} \sim q_{\mathcal{D}}(\tilde{x}|x)$ stochastic mapping. The basic autoencoder maps the corrupted input $\tilde{x}$ to the hidden representation $y = f_\theta(\tilde{x}) = s(W\tilde{x} + b)$ as does the basic autoencoder. What is the best way to reassemble $z = g_\theta'(y)$. Where, θ^* and θ' are trained to reduce the average reconstruction error on the training set, i.e. to get z as near to the uncorrupted input x as possible. The primary difference is that instead of x, z i is now a deterministic function of $\tilde{x}$. The reconstruction error is the square of the error loss $L_2(x, z) = ||x - z||^2$ with affine decoder, as before. The parameters are randomly initialized before being optimized using a descending stochastic gradient. Note that each time a training example x is shown, $q_{\mathcal{D}}(\tilde{x}|x)$ generates a corrupted version that differs $\tilde{x}$ from it.

3.3 Stacked denoising autoencoder

The denoising autoencoder is layer-by-layer layered in the form of a deep network structure for more sophisticated feature expression. The model structure is then built using the stacked denoising autoencoder, which links the denoising autoencoder up and down. The previous output is pure input later in the training process, followed by layer-by-layer training. Fig. 3 depicts the training procedure.

Stacking denoising autoencoders is demonstrated in Fig. 3. The learnt encoding function f_θ is applied to the clean input after training the first level

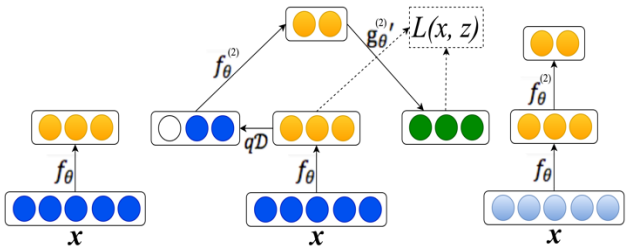

Figure. 3 Stacked denoising autoencoder

denoising autoencoder (shown in Fig. 3) (left). The second (middle) level denoising autoencoder is trained to learn the second level encoding function $f_\theta^{(2)}$ using the generated representation. The procedure can then be repeated (right).

3.4 Denoising autoencoder with dropout regularization

To create a deep neural network using an unsupervised method, DAE was employed as the deep neural network's hidden layer. A huge number of *x* inputs are employed to initialize the model parameters in our paper. The initialization method on the multi-layer encoder model is obtained via layer-by-layer DAE training. The output of the lower layer DAE is the input of the upper layer. The DAE for all layers is a deep neural network (DNN) hidden layer structure. During this phase, there may be concerns with network overfitting.

Overfitting is a difficult problem to solve when building high-dimensional deep learning models, and it must be treated with caution. A recently discovered regularization strategy known as "dropout" has shown to be quite successful in neural networks [34]. Dropout is an approach for reducing "overfitting", while training neural networks with limited training data sets [35]. Furthermore, the dropout approach can avoid repeating features caused by DAE hidden layer mutual adaptation [36].

To create neural networks, dropout and dropconnect techniques are utilized [26], [37]. A fully connected layer is the most basic component of a neural network, consisting of a linear translation of the input vector *z* to the output vector *x*, with nonlinearity applied to the component *x*. Given a generic linear transformation *x* = *Gz* with column vectors *z* and *x*, the regularization approach is based on multiplying the components *z* (dropout) and *G* (dropconnect) by an independent bernoulli random variable. As a result, the *x* component is computed as follows [38]:

$$x_i = \sum_k g_{ik}(\xi_k z_k) => dropout \quad (5)$$

$$x_i = \sum_k (\xi_{ik} g_{ik}) z_k => dropconnect \quad (6)$$

Where, $\xi_k, \xi_{ik} \sim \mathcal{B}(1-\mathrm{p})$ with the hyperparameter *p* known as the rate of decline. The element x_i is approximately Gaussian for Lyapunov's central limit theorem (Wang and Manning, 2013), so that the distribution as:

$$x_i \sim \mathcal{N}(\sum_k \theta_{ik}; \alpha \sum_k \theta_{ik}^2) \quad (7)$$

Where, $\alpha = p/(1-p)$ and $\theta_{ik} = g_{ik} z_k (1-p)$.

Unit dropout, in particular, removes the unit from the network, as well as all incoming and outgoing links within the network. The most basic technique is to store each unit in the network with a retention probability p that is independent of the other units. The probability p can either be chosen from the validation set or set naively to 0.5. Although simple, the pre-set probability p at 0.5 looks to be near ideal for various networks and applications. One example is that the best probability of retention for input units is frequently closer to 1 than 0.5 [39].

4. Materials and methods

We offer a deep learning methodology (autoencoder family) for imputing missing load data in this part:

4.1 Problem statement

The dataset is defined as $\mathcal{D} = \{(X_n, y_n)\}_{n=1}^N$, $X \in \mathbb{R}^d$ for $d = 1,2,\dots,d$, for a basic matrix consisting of n samples and d features. Where, $X_n = (x_i^1, x_i^2, \dots., x_i^d) \in \mathbb{R}^d$ and $y \in \mathbb{R}^1$ is an effort to develop this software. In this work, our dataset contains several sets of categorical feature indices (converting numeric using ordinal encoding), with features taking discrete (continuous) values. The goal is to keep losses and metrics simple, so that the focus is on data sets that have become numerical features only.

The data set with missing values will be assigned to X. Thus, it can be defined in the missingness matrix for, $M(0,1)^{n \times d}$ such that x_i^j is missing if and only if $m_i^j = 0$. We state, that $\mathcal{D}^* = (X_n^*, y), X_n^* \in \mathbb{R}^d$ is a basic truth dataset without missing data. Although, the dataset that we use is only one dataset that has more than 40 % missing data, namely ISBSG10. We attempted to retrieve data using MCAR in this investigation. The use of dataset $\mathcal{D}$ by determining the appropriate missingness matrix *M* (range missingness used is 10 % to 80 %). Thus, the imputation function is defined as: $f: \mathbb{R}^d \times (0,1)^d \to \mathbb{R}^d = (x, m) \to f'(x, m)$. The purpose of implementing this function (f') is to minimize reconstruction metrics in missing data imputation.

4.2 Development of the imputation approach via SDAE-dropout regularization

Deep learning and neural networks (DNN) are new advances in machine learning that can handle large volumes of data and learn high-level representations. In general, when a neural network

(NN) model works on a known small training data set, there will be an overfitting problem [20]. According to Hinton et al (2012), when training NN with a small training dataset, using the dropout strategy can assist reduce overfitting [35]. Dropout is produced by turning off the output of some hidden neurons, preventing them from taking part in the forward propagation training process. Dropout is turned off during testing, implying that all hidden neurons' output is visible [20].

Denoising autoencoder (DAE) has recently attracted the attention of the research community due to its nature in terms of the ability to learn from corrupted data, which is an extension to missing data fields [3, 40]. DAE, on the other hand, can corrupt a section of the input data with stochastic noise injection and then attempt to reconstruct it using nested nonlinear transformations [19].

We present a stacked denoising autoencoder (SDAE) based model of missing data imputation for regression models with small data sets, motivated by the reported inadequacies of applying the denoising autoencoder (DAE) to the problem of missing data imputation (shown Fig. 4). Thus, our research intends to capture hidden correlations between missing and non-missing values in small data sets utilizing SDAE based on dropout regularization approaches, and then estimate for imputations. Dropout regularization approach is used in both the input and hidden layers to avoid overfitting during fine-tuning. If necessary, we also allow L_2 regularization during training to further limit the risk of model overfitting. In addition, the proposed SDAE is a model that is driven by missing values with model training process and a different imputation strategy in the regression field.

In this paper, we will compare our proposed method with four advanced missing imputation techniques, including GAIN (generative adversarial imputation networks) [13], MICE (multiple imputation by chained equations) [28], MissForest (random forest imputation) [27], and k-nearest neighbor imputation (kNNI) [29].

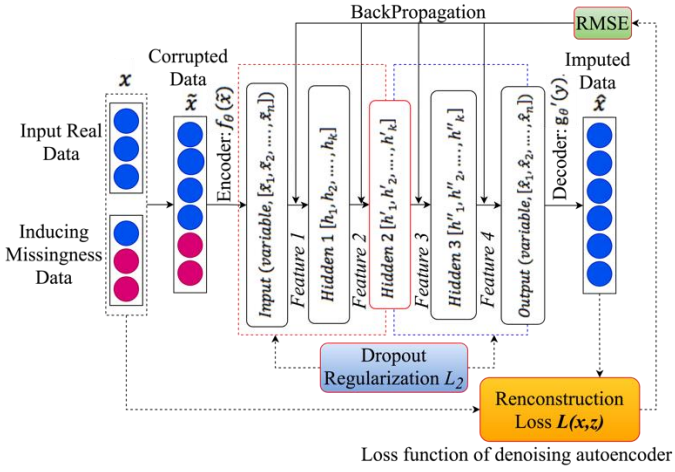

Figure. 4 Our proposed improved SDAE with dropout regularization

4.3 Data collection and data preprocessing

The most widely used dataset related to the SEE context is the repository in PROMISE (PREdictOr models in software engineering) and ISBSG, which is one of the most popular datasets [41–44]. In the early 2000s, the usage of small datasets was quite popular, and most empirical research in software engineering was done using small samples [12]. Because gathering and reporting data from the project is expensive, the development team focused less on data collection [9]. This causes incomplete datasets to appear frequently throughout SEE studies on that dataset.

The data set used in our paper is shown in Table 1, which includes the number of projects, features, categorical features, predictor feature name, and target feature name. PROMISE is an online data repository that is open to the public. We used 9 datasets available in the promise repository, such as: maxwell, cocomo81, kitchenham, nasa93, kemerer, albrecht, desharnais, China and the preprocessing rules used in the study by [45, 46], and UCP dataset as per rules [47]. Meanwhile, 2 datasets are available from ISBSG such as: ISBSG18 refers to research [48], and ISBSG10 refers to research [49].

Data preprocessing is conducted out after data collection in order to improve data quality. Feature reduction, or deleting irrelevant features, is frequently part of the data pre-processing procedure. The next step is to convert categorical data into numeric using ordinal encoding. Since each category is displayed as a single input, the advantage is that the dimensions of the problem space do not increase [50]. Give the *i*-th object the value f and f has a status sorted M_f with rank $\{1,2,\ldots,M_f\}$. $r_{if} = \{1,2,\ldots,M_f\}$ should be used to replace each x_{if}. So that each attribute has the same weight, change the range of each attribute to [0, 1]. Use z_{if} to display the *i*-th object's attributes r_{if}.

$$z_{if} = \frac{r_{if}-1}{M_f-1} \tag{8}$$

Data normalization (DN) changes the value of a feature based on predefined rules to ensure that all scaled features have the same impact [51]. The interval [0, 1] will be used as a scaling goal in our study, as illustrated in Eq. (9).

Table 1. The study's dataset

Dataset	Criterion					
	ID	***Proj***	***N***	***Cat.***	***Predictors***	***Target***
Without missing values:						
China	Ch	499	17	0	AFP, Input, Output, Enquiry, File, Interface, Added, Changed, Deleted, PDR_AFP, PDR_UFP, NPDR_AFP, NPDU_UFP, Resource, Dev.Type, Duration	Effort
Albrecth	Al	24	7	0	Input, Output, Inquiry, File, AdjFP, RawFPcounts, Effort	Effort
Maxwell	Mw	62	26	0	App, Har, Dba, Ifc, Source, Telonuse, Nlan, T01, T02, …., T14, T15, Duration, Size, Time	Effort
Nasa93	N93	93	18	16	mode, rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced, equivphyskloc	act_effort
Cococmo81	C81	63	17	0	rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced, loc	Actual
Kitchenham	Kt	145	4	0	Actual.duration, Adjusted.function.points, First.estimate	Actual.effort
Kemerer	Km	16	6	0	language, hardware type, estimated duration, AdjFP, RawFP	KSLOC
Desharnais	Dh	81	8	0	Team-Exp, ManagerExp, Transactions, Entities, PointsNonAdjust, Adjustment, Language	PointsAjust
UCP	Ucp	71	5	0	UAW, UUCW, TCF, ECF	Real_Effort_Person_Hours
With missing values:						
ISBSG10	I10	952	11	6	FunctionalSize, ValueAdjustmentFactor, ProjectElapsedTime, DevelopmentType, BusinessAreaType, ClientServer, DevelopmentPlatform, LanguageType, FirstOS, MaxTeamSize	NormalisedWorkEffortLevel1
ISBSG18 IFPUG	I18	36	12	11	Data_Quality, UFP, IS, DP, LT, PPL, CA, FS, RS, Recording_Method, FPS	S_effort

$$x_k \; in \; [0,1] = \frac{x_k - x_{min}}{x_{max} - x_{min}} \tag{9}$$

Where, x is the data matrix's feature column. x_k corresponds to the k-th value in x, while x_{min} and x_{max} relates to the minimum and maximum values in x. The mean and standard deviation of x are used to determine $\bar{x}$ and $std(x)$.

Finally, the data was normalized using the Z-score normalization technique. It can also be used to eliminate data outliers in order to improve data quality [52]. The unstructured data can then be standardized with the Z-score parameter, according to Eq. (10) [53]:

$$Z - score \; x_k = \frac{x_k - \bar{x}}{std(x)} \tag{10}$$

4.4 Performance analysis

Fig. 5 shows the model training and validation procedures. The first stage is data preparation, which is then followed by a preprocessing process consisting of feature reduction, categorical conversion, normalization, and the sequence of available data from each building is randomly partitioned into training, validation, and testing data sets with a proportion of each 70 %, 15 %, and 15 %. After that do inducing missingness (10 %-80 %). The next step is to process missing data using SDAE-dropout regularization. Finally, carry out the process of evaluating the imputed data.

A widely used performance metric in the software effort estimation literature is the measured error rate of estimation [54], [55]. The metrics utilized in evaluation are mean absolute error (MAE) and root mean square error (RMSE). It's better if the model's MAE and RMSE values are low.

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |X_i - Y_i| \tag{11}$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2} \tag{12}$$

5. Evaluation and results analysis

In this section, we present the results of an experimental assessment of the proposed method's

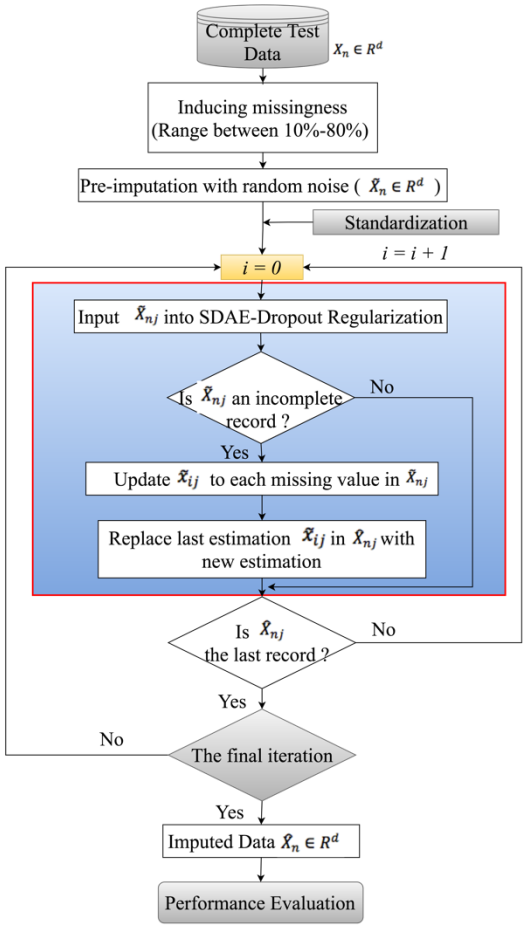

Figure. 5 Flowchart of our model training scheme

effectiveness. Experiments were carried out using a computing platform based on Intel Xeon Quad-core 2.4 GHz CPU, GeForce GPU Titan 100, memory 64 Gb and Microsoft Windows 8 R2 Professional 64-bit. The development environment is notepad plus, text editor/IDE, anaconda web programming interface, several libraries on Scikit-learn, and NumPy. Default parameter settings provided by python 3.0 are used.

5.1 Hyperparameter setting

The set of hyperparameters in the proposed SDAE-dropout regularization model will perform optimally for a particular problem, and varies with the amount of data set corrupt. The hyperparameters set in the model for all trained benchmarks, we use the embedding dimension of the number hidden layer of 128 enough for an overly complete representation of all the data sets involved, which aims to improve the generalization performance of the DNN. DNN has the capability of automatically extracting features, resulting in a decrease in the number of hidden units as the value increases. Next, we train the model for a maximum of 500 iterations with an early stop strategy for reconstruction losses over known elements. Because in our study, using a small data set, we added the technique of regularization (L_2) parameters of 1×10^{-4} and dropout probabilities of 0.5 were used to avoid overfitting problems. If L_2 is very large, however, it will add too much weight and produce a lack of fit. Simple 3-layer feed-forward network trained 10 times for each autoencoder optimization step. In TensorFlow, the Adam optimizer default learning rate for both networks is 1×10^{-3} (1e-3) each for SDAE. We examine ReLU as a hidden layer activation function to tackle the missing gradient problem generated by Sigmoid or the explosion gradient problem created by ReLU [56], [57]. Also, we apply a random Gaussian noise of 0.2 for each time step of the training data progression in overcoming the noise data during the training process [58].

We compared four advanced techniques, including GAIN [13], MICE [28], MissForest [27], and kNNI [29], to validate our methodology. Whereas, for the comparison method we used, the adjustment for the imputation approach i.e., the kNNI parameter k was set as 5 because the imputation error was low in the acceptable time range for all three data sets achieved with k = 5. Because it reconstructs the complete data set at once, MissForest focuses on the repair data set. MissForest employs a total of 100 estimators, each with no leaf-node limit and a maximum of ten iterations (n=10). MICE and GAIN have the same maximum number of iterations as MissForest.

5.2 Comparison algorithms

There are only two data sets in the SEE context that are induced by missing values, while other data without have missing values. So, in this study we will apply inducing missingness with a range between 10 % to 80 % in our algorithm and the comparison algorithm. Consider a dataset with a $\mathcal{D}$-dimensional object x where each feature (which is denoted by x_i) may be missing and the target value is y. Missing values in objects are not supported by the majority of discriminating approaches. Missing features imputation is the process of filling in missing feature values.

We standardized the numerical data using min-max normalization and utilized ordinal encoding to express the categorical data as model parameters for ease of evaluation and faster convergence. Furthermore, we repeat each test ten times and publish the average results in order to achieve valid experimental data.

Using multiple publicly accessible real-world datasets on PROMISE and ISBSG, we analyze the

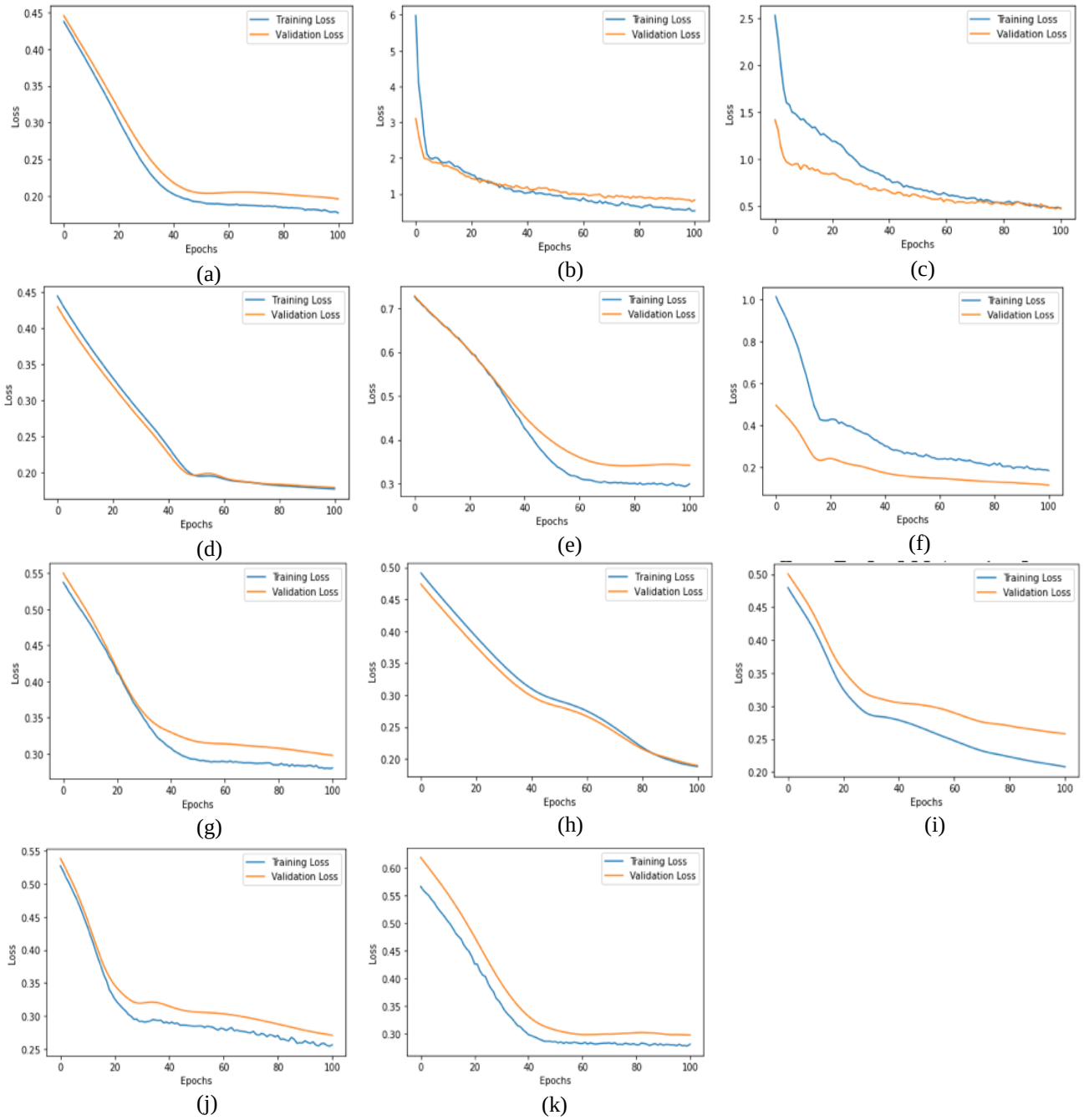

Figure. 6 The comparison of validation loss of our models (training phases): (a) Albrecht dataset, (b) China dataset, (c) Cocomo81 dataset, (d) Desharnais dataset, (e) ISBSG IFPUG dataset, (f) ISBSG10 dataset, (g) Kemerer dataset, (h) Kitchenham dataset, (i) Maxwell dataset, (j) Nasa93 dataset, and (k) UCP dataset

imputation quality achieved by the performance of SDAE-dropout regularization in this section. Before training, we dropped 70 % of the good values on the train, 15 % for the test set, and 15 % for the validation test set at random.

Our SDAE will estimate the missing values as closely as feasible to the real data once we partition the dataset into three parts. At the start of each epoch, we generate a new corruption vector based on the Gaussian distribution by multiplying the original data by a sample threshold in the center of the distribution to create a corrupted input for the encoder. To minimize the modified loss function, we use 3-fold cross-validation and the dropout regularization approach to train our model. Fig. 6 shows that by applying the SDAE method, the missing data can be estimated well. This shows that our proposed model does not experience overfitting problems or expensive computations. These findings show that the SDAE model reduces the predicted loss over the empirical distribution of not just the observed data

but also a portion of the previously seen corrupted data, hence lowering the chance of bias.

Because there are so many different types of parameters, using a single autoencoder learning method will result in large processing costs and a loss of learning efficiency [17]. As a result, customization necessitate a computationally efficient approach. In this instance, the training data must be reconfigured based on the workload by taking into account numerous parameters when constructing the autoencoder. By applying the dropout technique to achieve better results than non-dropout. The application of the dropout technique is not only faster convergence for the encoder decoder to study the regression data pattern only, but also achieves better performance in preventing overfitting will be obvious if the NN is deeper.

On a given data set, we will compare the imputation performance of our technique to the four initial imputation methods based on the accuracy of the missing data estimate in the next stage. The GAIN, kNNI, MICE, and MissForest approaches are among the ones we compare to ours. As is customary, we employ MAE and RMSE as performance measurements between the baseline truth and approximation values. Note that in this series of studies, additional faults were introduced to the data set by utilizing the MCAR missing generation approach to remove 80 percent of all data points at random. The comparison results of our imputation approach using MAE and RMSE are shown in Tables 2 and 3.

Table 2. Comparison imputation performance using MAE (with missing rate 80 %)

ID	Our model	GAIN	kNNI	MICE	MissRF
Al	**0.1713**	0.2437	0.3405	*0.5944*	0.3126
Ch	0.1254	*0.1263*	**0.0564**	0.0892	0.0661
C81	**0.1281**	0.1362	0.2621	*0.3512*	0.2375
Dh	**0.1422**	0.1425	0.1907	*0.2567*	0.1902
I18	**0.1989**	0.2489	0.2741	*0.4129*	0.2948
I10	**0.1061**	0.1063	0.1902	0.2474	*0.3015*
Km	0.1711	**0.0279**	0.3010	*0.3862*	0.2799
Kt	*0.1215*	**0.0338**	0.0865	0.0918	0.0850
Mw	**0.1766**	0.1826	0.3170	*0.3735*	0.2916
N93	**0.1591**	0.1675	*0.3039*	0.2757	0.2470
Ucp	0.1074	*0.1695*	**0.1033**	0.1219	0.1190

Table 3. Comparison imputation performance using RMSE (with missing rate 80 %)

ID	Our model	GAIN	kNNI	MICE	MissRF
Al	**0.3273**	0.3697	0.3767	*0.9320*	0.3550
Ch	**0.0841**	0.0947	0.0990	*0.1638*	0.1139
C81	**0.2792**	0.3931	0.3235	*0.4881*	0.3138
Dh	**0.2256**	0.2552	0.2364	*0.3609*	0.2375
I18	**0.2428**	*0.5034*	0.2819	0.5310	0.3652
I10	**0.2149**	0.2840	0.2278	*0.4435*	0.3814
Km	**0.1467**	0.1515	0.2836	*0.4230*	0.2806
Kt	**0.0839**	*0.1392*	0.0916	0.0932	0.0905
Mw	**0.2901**	*0.4760*	0.3839	0.4604	0.3830
N93	**0.2854**	*0.4661*	0.3127	0.3109	0.3016
Ucp	**0.1025**	*0.2939*	0.0976	0.1065	0.1097

The goal of the imputation method is to reliably restore missing data (from minor to severe corruption). As a result, we anticipate that after using this strategy, the broken distribution will be forced to follow the true distribution. Bold numbers indicate the highest accurate imputation; italics, on the other hand, indicate the least accurate imputation. Based on the results in the table using a data missing rate of 80 %, showing the MAE value (Table 2), that our model produces the poor error in the kitchenham dataset with a value of 0.1215 and is followed by the china dataset (0.1254), UCP (0.1074), and kemerer (0.1711). Meanwhile, in other datasets our model produces the best performance. In Table 3, which shows the performance of our model using the RMSE value has the best performance on all data sets. This shows that the use of SDAE can overcome the large number of calculations required, local minimums, and missing gradient problems [17]. When using backpropagation, the use of numerous propagation and optimizer functions makes learning easier [59], and the addition of a dropout technique helps reduce overfitting when training a limited dataset [35]. This is why our model has the best accuracy rate than some of the other popular methods used in our study.

As for the comparison method, based on the results in Tables 2 and 3, it shows that GAIN has an MAE value with the poor performance in the China dataset (0.1263) and UCP (0.1695). Meanwhile, GAIN has the best performance on the kemerer (0.0279) and kitchenham (0.0338) datasets. However, GAIN also has the poor RMSE value in the datasets kitchenham (0.1392), UCP (0.2939), Nasa93 (0.4661), maxwell (0.4760), and ISBSG18 (0.5034). This is because GAIN is a modeling that has a complex network and difficult training. If it is run on a small dataset, it can cause overfitting and bias in the imputation results. This technique, according to Nazabal et al (2018), can only handle continuous or binary data, and adapting to diverse data is difficult. As a result, there is still a need for approaches for efficiently training deep generative models on incomplete and diverse data sets [60].

kNNI has the best performance as indicated by the MAE value in the china dataset (0.0564) and UCP (0.1033), but has the poor value in the nasa93 dataset (0.3039). MICE most of the datasets produce the poor

Figure. 7 RMSE values obtained from five imputation methods: (a) Albrecht dataset, (b) China dataset, (c) Cocomo81 dataset, (d) Desharnais dataset, (e) IFPUG dataset, (f) ISBSG10 dataset, (g) Kemerer dataset, (h) Kitchenham dataset, (i) Maxwell dataset, (j) Nasa93 dataset, and (k) UCP dataset

performance. Meanwhile, MissForest only has one poor performance on the MAE value in the ISBSG10 dataset (0.3015). This shows that MICE, MissForest and kNNI are data imputation methods that do not have resistance to noise which can cause low accuracy values. kNNI, on the other hand, cannot be employed at greater missing rates due to the limitation that it can only use complete cases as probable nearest neighbors (as seen in this experiment). MICE and MissForest experienced the similar problem, resulting in poor utility when dealing with small data and noise. Whereas, our model's improved performance in this case with a small data set size and constrained dimensions demonstrates its utility when faced with high dimensional missing values, which is a performance barrier for some other imputation algorithms, whereas our model can tolerate noise. Meanwhile, for small datasets, the computational costs of our model are comparable to or better than imputation using GAIN, kNNI, MICE, and MissForest.

5.3 Missingness sensitivity

We will assess our algorithm in this section to see how sensitive it is to a specific missing percentage ranging from 10 % to 80 %. We also compare this strategy to a few other imputation methods in this section. It is important to note that our proposed solution successfully compensates for the vast majority of situations, especially those with missing intervals of up to 80 %. These are the most severe circumstances, due to the high level of corruption, estimation of missing data becomes inconsistent and impossible to correct using other approaches. Each data set's feature is successively injected with missing and calculated data, with imputation quality evaluated by comparing the underlying truth and calculated data using the root mean square error (RMSE) measure. The association between missing data rate and imputation performance for the deleted data period was explored and illustrated in Fig. 7 to further analyze the value of the offered technique.

When utilizing a conventional training scheme based on imputation methods, the five polylines in each subgraph show essentially identical patterns of variance, indicating that the rationality of the original approximation of missing values has a direct impact on imputation accuracy. Furthermore, when the amount of missing data grows, the trend of variance and polyline size in our technique and kNNI are nearly identical. Nonetheless, the RMSE values achieved by our technique are marginally better than those obtained by kNNI in the majority of cases. For larger missing rates, our technique was more effective, followed by kNNI, which outperformed standard MICE, MissForest, and GAIN. Unfortunately, our method's performance on the China dataset has a low value trend. kNNI, MICE, and MissForest all performed similarly, with deteriorating estimating results and a greater data rate missing rate of more than 70 %. In comparison to previous algorithms, kNNI and MissForest provide steady performance for 20 % to 70 % missing values based on low RMSE values. Finally, our technique reveals that all missing rates have identical performance trends. By giving the best imputation methodology, which surpasses all other methods for missing data rates ranging from 10 % to 80 %. As a result, the proposed SDAE with dropout regularization provides a more accurate approximation of the actual measurement for majority-omitted interval imputation at various missing levels.

6. Conclusion

This study introduces a new approach to missing data imputation called stacked denoising autoencoder with dropout regularization technique. Our model is trained using selected parameters to estimate missing data on a small data set to tackle the regression challenge. In an experiment with 11 datasets from clinical trials injected with missing values under the MCAR, the methodology was compared to four state-of-the-art algorithms (GAIN, kNNI, MICE, and MissForest) (including 10 percent to 80 percent missing rates). Our solution exceeds all others in terms of total improvement and low error rate.

Due to the high cost and difficulty of obtaining data, eliminating cases with missing data can result in a sample size that is too small, resulting in bias. We use DAE, which consistently accounts for missing data with minimal error, even for data sets containing many missing data at the start. Meanwhile, in our approach, the Stacked mechanism tries to limit process diversity and the influence of corruption. The dropout regularization technique is used to limit the number of neurons in the layer, which helps to avoid overfitting and the disappearance of gradients when the model is trained. In the missing imputation model, this might lead to biased results. We discovered that going through the standardization stage can help cut optimization time and prevent overfitting for small datasets. Experiments on eleven data sets showed that our technique is effective, particularly for significant corruption volumes and mixed data.

New studies will be done in the future to test the SDAE technique with datasets from other contexts,

some of which contain missing values already, and to apply the remaining missing data procedures (MNAR and MAR). Furthermore, investigating the impact of imputation in the classification task is a crucial aspect.

Conflicts of interest

There are no conflicts of interest declared by the authors.

Author contributions

Conceptualization, R. Marco; methodology, R. Marco, S. S. S. Ahmad and S. Ahmad; validation, S. S. S. Ahmad and S. Ahmad; formal analysis, R. Marco, S. S. S. Ahmad and S. Ahmad; investigation, R. Marco; resources, R. Marco, S. S. S. Ahmad and S. Ahmad; data curation, R. Marco; writing—original draft preparation, R. Marco; writing—review and editing, S. S. S. Ahmad and S. Ahmad; visualization, R. Marco; supervision, S. S. S. Ahmad and S. Ahmad; funding acquisition, R. Marco.

References

[1] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Research patterns and trends in software effort estimation", *Journal of Information and Software Technology*, Vol. 91, pp. 1–21, 2017.

[2] F. A. Amazal, A. Idri, and A. Abran, "Software development effort estimation using classical and fuzzy analogy: A cross-validation comparative study", *International Journal of Computational Intelligence and Applications*, Vol. 13, No. 3, pp. 1–19, 2014.

[3] L. Gondara and K. Wang, "MIDA: Multiple imputation using denoising autoencoders", *Journal of Advances in Knowledge Discovery and Data Mining*, Vol. 10939, pp. 260–272, 2018.

[4] I. Abnane, A. Idri, and A. Abran, "Fuzzy case-based-reasoning-based imputation for incomplete data in software engineering repositories", *Journal of Software: Evolution and Process*, Vol. 32, No. 9, pp. 1–25, 2020.

[5] I. Abnane, M. Hosni, A. Idri, and A. Abran, "Analogy Software Effort Estimation Using Ensemble KNN Imputation", In: *Proc. of International Conf. On Software Engineering and Advanced Applications, Kallithea, Greece*, pp. 228–235, 2019.

[6] R. J. A. Little and D. B. Rubin, "The Analysis of Social Science Data with Missing Values", *Journal of Sociological Methods and Research*, Vol. 18, No. 2–3, pp. 292–326, 1989.

[7] J. Li, A. A. Emran, and G. Ruhe, "Impact Analysis of Missing Values on the Prediction Accuracy of Analogy-based Software Effort Estimation Method AQUA", In: *Proc. of International Conf. on Empirical Software Engineering and Measurement (ESEM)*, Madrid, Spain, pp. 126–135, 2007.

[8] A. Idri, I. Abnane, and A. Abran, "Missing data techniques in analogy-based software development effort estimation", *Journal of Systems and Software*, Vol. 117, pp. 595–611, 2016.

[9] I. Myrtveit, E. Stensrud, and U. Olsson, "Assessing the benefits of imputing ERP projects with missing data", In: *Proc. of International Software Metrics Symposium*, pp. 78–84, 2001.

[10] M. L. Yadav and B. Roychoudhury, "Handling missing values: A study of popular imputation packages in R", *Journal of Knowledge-Based Systems*, Vol. 160, pp. 104–118, 2018.

[11] S. Tihon, M. U. Javaid, D. Fourure, N. Posocco, and T. Peel, "DAEMA: Denoising Autoencoder with Mask Attention", In: *Proc. of International Conference on Artificial Neural Networks*, Vol. 12891, pp. 229–240, 2021.

[12] A. Idri, I. Abnane, and A. Abran, "Systematic mapping study of missing values techniques in software engineering data", In: *Proc. of International Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Takamatsu, Japan, 2015.

[13] J. Yoon, J. Jordon, and M. V. D. Schaar, "GAIN: Missing data imputation using generative adversarial nets", In: *Proc. of International Conference on Machine Learning*, ICML 2018, Vol. 13, pp. 9042–9051, 2018.

[14] P. Yin and J. Q. Shi, "Simulation-based sensitivity analysis for non-ignorably missing data", *Statistical Methods in Medical Research*, Vol. 28, No. 1, pp. 289–308, 2019.

[15] T. Huamin, D. Qiuqun, and X. Shanzhu, "Reconstruction of time series with missing value using 2D representation-based denoising autoencoder", *Journal of Systems Engineering and Electronics*, Vol. 31, No. 6, pp. 1087–1096, 2020.

[16] J. Zhang and P. Yin, "Multivariate Time Series Missing Data Imputation Using Recurrent Denoising Autoencoder", In: *Proc. of IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2019*, pp. 760–764, 2019.

[17] J. C. Kim and K. Chung, "Multi-Modal Stacked Denoising Autoencoder for Handling Missing Data in Healthcare Big Data", *IEEE Access*, Vol.

8, pp. 104933–104943, 2020.
[18] F. M. Bianchi, L. Livi, K. Ø. Mikalsen, M. Kampffmeyer, and R. Jenssen, "Learning representations for multivariate time series with missing data using Temporal Kernelized Autoencoders", arXiv preprint, pp. 1–18, 2018.
[19] R. Lall and T. Robinson, "Applying the MIDAS Touch : An Accurate and Scalable Approach to Imputing Missing Data", *APSA Preprints*, 2020.
[20] J. Yu, X. Zheng, and J. Liu, "Stacked convolutional sparse denoising auto-encoder for identification of defect patterns in semiconductor wafer map", *Computers in Industry*, Vol. 109, pp. 121–133, 2019.
[21] K. Wang, P. Guo, X. Xin, and Z. Ye, "Autoencoder, low rank approximation and pseudoinverse learning algorithm", In: *Proc. of IEEE International Conference on Systems*, Man, and Cybernetics, SMC 2017, pp. 948–953, 2017.
[22] P. Vincent and H. Larochelle, "Extracting and Composing Robust Features with Denoising Autoencoder", In: *Proc. of the 25th International Conference on Machine Learning*, pp. 1096–1103, 2008.
[23] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models", *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.
[24] C. Jia, M. Shao, S. Li, H. Zhao, and Y. Fu, "Stacked Denoising Tensor Auto-Encoder for Action Recognition with Spatiotemporal Corruptions", *IEEE Transactions on Image Processing*, Vol. 27, No. 4, pp. 1878–1887, 2018.
[25] A. F. Costa, M. S. Santos, J. P. Soares, and P. H. Abreu, "Missing data imputation via denoising autoencoders: The untold story", *Springer Nature Switzerland*, Vol. 11191 LNCS, pp. 87–98, 2018.
[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, Vol. 15, pp. 1929–1958, 2014.
[27] D. J. Stekhoven and P. Bühlmann, "Missforest-Non-parametric missing value imputation for mixed-type data", *Journal of Bioinformatics*, Vol. 28, No. 1, pp. 112–118, 2012.
[28] S. V. Buuren and K. G. Oudshoorn, "mice: Multivariate imputation by chained equations in R", *Journal of Statistical Software*, Vol. 45, No. 3, pp. 1–67, 2011.
[29] I. Abnane and A. Idri, "Improved analogy-based effort estimation with incomplete mixed data", In: *Proc. of the 2018 Federated Conference on Computer Science and Information Systems*, FedCSIS 2018, Vol. 15, pp. 1015–1024, 2018.
[30] H. Lu, G. Perrone, and J. Unpingco, "Multiple Imputation with Denoising Autoencoder using Metamorphic Truth and Imputation Feedback", arXiv preprint, Vol. 2, 2020.
[31] S. Ryu, M. Kim, and H. Kim, "Denoising Autoencoder-Based Missing Value Imputation for Smart Meters", *IEEE Access*, Vol. 8, pp. 40656–40666, 2020.
[32] J. Chen and X. Shi, "Sparse convolutional denoising autoencoders for genotype imputation", *Genes*, Vol. 10, No. 9, pp. 1–16, 2019.
[33] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", *Journal of Machine Learning Research*, Vol. 11, pp. 3371–3408, 2010.
[34] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, and X. Chen, "A sparse auto-encoder-based deep neural network approach for induction motor faults classification", *Measurement: Journal of the International Measurement Confederation*, Vol. 89, pp. 171–178, 2016.
[35] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", arXiv preprint, pp. 1–18, 2012.
[36] Z. Chen and Z. Li, "Fault diagnosis method of rotating machinery based on stacked denoising autoencoder", *Journal of Intelligent and Fuzzy Systems*, Vol. 34, No. 6, pp. 3443–3449, 2018.
[37] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of Neural Networks using DropConnect", In: *Proc. of the 30th International Conference on Machine Learning*, Vol. 28, 2013.
[38] L. Antelmi, N. Ayache, P. Robert, and M. Lorenzi, "Sparse multi-channel variational autoencoder for the joint analysis of heterogeneous data", In: *Proc. of 36th International Conference on Machine Learning*, ICML 2019, Vol. June, pp. 453–464 , 2019.
[39] R. Xie, J. Wen, A. Quitadamo, J. Cheng, and X. Shi, "A deep auto-encoder model for gene expression prediction", *BMC Genomics*, Vol. 18, No. 9, 2017.
[40] A. F. Costa, M. S. Santos, J. P. Soares, and P. H. Abreu, "Missing data imputation via denoising autoencoders: The untold story", *Springer Nature Switzerland*, Vol. 11191, pp. 87–98, 2018.
[41] A. K. Bardsiri, S. M. Hashemi, and M. Razzazi,

"Statistical Analysis of the Most Popular Software Service Effort Estimation Datasets", *Journal of Telecommunication, Electronic and Computer Engineering*, Vol. 7, No. 1, pp. 87–96, 2015.

[42] B. Vasilescu, A. Serebrenik, and T. Mens, "A historical dataset of software engineering conferences", In: *Proc. of IEEE International Working Conference on Mining Software Repositories*, pp. 373–376, 2013.

[43] T. R. Benala, R. Mall, P. Srikavya, and V. HariPriya, "Software Effort Estimation Using Data Mining Techniques", *Advances in Intelligent Systems and Computing*, Vol. 248, pp. 85–86, 2014.

[44] D. Déry and A. Abran, "Investigation of the effort data consistency in the ISBSG repository", In: *Proc. of the 15th Intern. Workshop on Software*, No. June, 2005.

[45] L. Song, L. L. Minku, and X. Yao, "A novel automated approach for software effort estimation based on data augmentation", In: *Proc. of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Vol. 18, pp. 468–479, 2018.

[46] T. Xia, R. Krishna, J. Chen, G. Mathew, X. Shen, and T. Menzies, "Hyperparameter Optimization for Effort Estimation", arXiv preprint, Vol. 4, 2018.

[47] M. Azzeh, A. B. Nassif, and S. Banitaan, "Comparative analysis of soft computing techniques for predicting software effort based use case points", *IET Software*, Vol. 12, No. 1, pp. 19–29, 2018.

[48] L. V. Arias and C. Q. López, "Comparative study of random search hyperparameter tuning for software effort estimation", In: *Proc. of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering*, pp. 21-29, 2021.

[49] L. L. Minku and X. Yao, "Can Cross-company Data Improve Performance in Software Effort Estimation ?", In: *Proc. of the 8th International Conference on Predictive Models in Software*, pp. 69–78, 2012.

[50] E. F. Norris, S. Vahid, and C. Hand, "Evaluating the Impact of Categorical Data Encoding and Scaling on Neural Network Classification Performance: The Case of Repeat Consumption of Identical Cultural Goods", *Journal of Communications in Computer and Information Science*, Vol. 311, pp. 343–0352, 2012.

[51] L. Angelis and I. Stamelos, "A Simulation Tool for Efficient Analogy Based Cost Estimation", *Journal of Empirical Software Engineering*, Vol. 5, No. 1, pp. 35–68, 2000.

[52] K. K. R. Samal, K. S. Babu, and S. K. Das, "Temporal convolutional denoising autoencoder network for air pollution prediction with missing values", *Urban Climate*, Vol. 38, No. February, pp. 100872, 2021.

[53] J. Huang, Y. F. Li, J. W. Keung, Y. T. Yu, and W. K. Chan, "An empirical analysis of three-stage data-preprocessing for analogy-based software effort estimation on the ISBSG data", In: *Proc. of International Conf. on Software Quality*, Reliability and Security, Prague, Czech Republic, pp. 442–449, 2017.

[54] E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit, "An Empirical Validation of the Relationship Between the Magnitude of Relative Error and Project Size", In: *Proc. of the Eighth IEEE Symposium on Software Metrics*, 2002.

[55] P. Phannachitta and K. Matsumoto, "Model-based software effort estimation - A robust comparison of 14 algorithms widely used in the data science community", *International Journal of Innovative Computing, Information and Control*, Vol. 15, No. 2, pp. 569–589, 2019.

[56] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network", *Sensors (Switzerland)*, Vol. 19, No. 11, 2019.

[57] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10", Unpublished manuscript, pp. 1–9, 2010.

[58] C. Fan, M. Chen, R. Tang, and J. Wang, "A novel deep generative modeling-based data augmentation strategy for improving short-term building energy predictions", *Building Simulation*, Vol. 15, No. 2, pp. 197–211, 2022.

[59] F. Gu, K. Khoshelham, S. Valaee, J. Shang, and R. Zhang, "Locomotion Activity Recognition Using Stacked Denoising Autoencoders", *IEEE Internet of Things Journal*, Vol. 5, No. 3, pp. 2085–2093, 2018.

[60] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera, "Handling incomplete heterogeneous data using VAEs", *Pattern Recognition*, Vol. 107, No. 107501, 2020.