# Imputation using training labels and classification via label imputation

Thu Nguyen
*Department of Holistic Systems*
*Simula Metropolitan*
Oslo, Norway
thu@simula.no

Pål Halvorsen
*Dept. of Holistic Systems*
*Simula Metropolitan*
Oslo, Norway
paalh@simula.no

Michael A. Riegler
*Dept. of Holistic Systems*
*Simula Metropolitan*
Oslo, Norway
michael@simula.no

***Abstract*—Missing data is a common problem in practical settings. Various imputation methods have been developed to deal with missing data. However, even though the label is usually available in the training data, the common practice of imputation usually only relies on the input and ignores the label. In this work, we illustrate how stacking the label into the input can significantly improve the imputation of the input. In addition, we propose a classification strategy that initializes the predicted test label with missing values and stacks the label with the input for imputation. This allows imputing the label and the input at the same time. Also, the technique is capable of handling data training with missing labels without any prior imputation and is applicable to continuous, categorical, or mixed-type data. Experiments show promising results in terms of accuracy.**

***Index Terms*—classification, missing data, imputation**

## I. Introduction

Data can come in various forms, ranging from continuous numerical values to discrete values and mixed between categorical and continuous. For example, a dataset on housing may contain information about the price, and areas, which are continuous, while the number of rooms, and number of bathrooms are discrete features. In addition, it is also common to encounter missing data. This can happen for various reasons, such as malfunctioning devices, broken sensors, or people declining to fill in survey information, etc. This can undermine the integrity and robustness of analyses, potentially leading to biased results and incomplete insights. Consequently, the need for methodologies to address missing data becomes paramount, prompting the widespread adoption of imputation techniques.

Imputation, as a remedial practice, involves the estimation or prediction of missing values based on observed information. This process becomes particularly intricate when dealing with datasets that encompass diverse data types, including continuous, categorical, and mixed data. In response to this complexity, a spectrum of imputation techniques has been developed, each tailored to handle specific data types and patterns [1], [2]. In addition, in recent years, classification or regression models that can handle missing data directly such as Decision Tree Classifier [3], LSTM [4] have also captured a lot of attention. However, imputation remained a popular choice as it renders the data complete for not only the classification or regression task but also data visualization or other inferences.

In this work, we propose stacking training labels to training input to improve the imputation of the training input and illustrate how this can significantly improve the imputation of training input. Moreover, we propose Classification Based on MissForest Imputation (CBMI) to predict the label on testing data without building a classification model on the training set. The method starts by initializing the predicted testing labels with missing values. Then, it stacks the input and the label, the training and testing data together, and uses missForest algorithm [2] to impute all the missing values in the matrix. The imputation returns a matrix that contains imputed training and testing input, imputed training labels if the training labels contain missing data, and predicted testing labels.

Our contribution is as follows: (i) We propose using training labels to aid the imputation of the training input and illustrate how this can significantly improve the imputation of training input; (ii) We propose CBMI algorithm for predicting the output of a classification task via imputation, instead of building a model on training data and then predicting on a test set; (ii) we conduct various experiments to illustrate that most of the time, the proposed approaches provide better accuracy for classification.

## II. Related works

A diverse array of techniques, ranging from traditional methods to sophisticated algorithms, has been developed to handle missing data. Each method brings unique strengths, making them suitable for different types of data and analysis scenarios. Traditional approaches, for example, complete case analysis, has been widely used due to its simplicity, but is prone to bias due to the exclusion of observations with missing values. More sophisticated methods, such as multiple imputation [5] or multiple imputation using Deep Denoising Autoencoders [6], have emerged to mitigate these challenges. The methods acknowledge the uncertainty associated with the imputation process by generating multiple plausible values for each missing data point. Another notable work is the Conditional Distribution - based Imputation of Missing Values (DIMV) [1] algorithm. The algorithm finds the conditional distribution of features with missing values based on fully observed features, and its imputation step provides coefficients with direct explainability similar to regression coefficients.

Regression and clustering-based methods also play a significant role in addressing missing data. The CBRL and CBRC algorithms [7], utilizing Bayesian Ridge Regression, showcase the efficacy of regression-based imputation methods. Additionally, the cluster-based local least square method [8] offers an alternative approach to imputation based on clustering techniques. For big datasets, deep learning imputation techniques are also of great use due to their powerful performance [9]–[11].

For continuous data, techniques based on matrix decomposition or matrix completion, such as Polynomial Matrix Completion [12], Alternating Least Squares (ALS) [13], and Nuclear Norm Minimization [14], have been employed to make continuous data complete, enabling subsequent analysis using regular data analysis procedures. For categorical data, various techniques have been explored. The use of k-Nearest Neighbors imputation involves identifying missing values, selecting neighbors based on a similarity metric (e.g., Hamming distance), and imputing missing values by a majority vote from the neighbors' known values. Moreover, the missForest imputation method [2] has demonstrated efficacy in handling datasets that comprise a mix of continuous and categorical features. In addition, some other methods that can handle mixed data include FEMI [15], SICE [16], and HCMM-LD [17].

In recent years, there has been a notable shift towards the development of classification methods and models capable of directly handling missing data, circumventing the need for imputation. These approaches recognize that imputation introduces a layer of uncertainty and potential bias and instead integrates mechanisms to utilize the available information effectively. Typically tree-based methods that have such capabilities are Gradient-boosted trees [18], Decision Tree Classifier [3], and their implementation are readily available in the sklearn package [19].

The Random Forest classifier has been a pivotal contribution in the field of machine learning, renowned for its robustness and versatility. Originating from ensemble learning principles, Random Forests combine the predictions of multiple decision trees to enhance overall predictive accuracy and mitigate overfitting. This approach was introduced by Breiman in 2001 [20], and since then, it has gained widespread adoption across diverse domains. The classifier's ability to handle high-dimensional data, capture complex relationships, and provide estimates of feature importance has made it particularly valuable. Various extensions and modifications to the original Random Forest algorithm have been proposed to address specific challenges. For instance, methods like Extremely Randomized Trees [21] introduce additional randomness during the tree-building process, contributing to increased diversity and potentially improved generalization. Additionally, research efforts have explored adapting Random Forests for specialized tasks, such as imbalanced classification, time-series analysis, and feature selection.

## III. Preliminary: missForest algorithm

MissForest is an imputation algorithm used for handling missing data in datasets. It is suitable not only for continuous or categorical but also for mixed-type data. The algorithm works by iteratively constructing a random forest based on the observed data, and then using this forest to predict missing values in the dataset. The process is repeated until convergence, refining the imputations in each iteration. MissForest is advantageous for its ability to handle non-linear relationships and interactions in the data, making it suitable for a wide range of applications. In addition, one key feature of MissForest is its adaptability to various types of missing data patterns, such as missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). This flexibility makes it a versatile tool for imputing missing values in datasets with complex structures.

**Algorithm 1 missForest algorithm**

**Input:**

1) a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$,
2) stopping criteron $\gamma$

**Procedure:**

initialize missing values in $\mathbf{X}$ by some imputation method
$\mathbf{k} \leftarrow$ sorted indices of features in $\mathbf{X}$ in an increasing order of missing values
**while not** $\gamma$ **do**:
    $\mathbf{X}_{old}^{imp} \leftarrow$ previously imputed version of $\mathbf{X}$,
    **for** $s$ in $\mathbf{k}$ **do**:
        fit a random forest where the observed values in $\mathbf{X}_s$ is the response, and the corresponding values of features in $\mathbf{X}$ other than $\mathbf{X}_s$ as the input
        predict $\mathbf{y}_{miss}^{(s)}$ (the missing values in $\mathbf{X}_s$) based on the features other $\mathbf{X}_s$.
        $\mathbf{X}_{new}^{imp} \leftarrow$ updated version of $\mathbf{X}$ using $\mathbf{y}_{miss}^{(s)}$
    **end for**
    Update $\gamma$.
**end while**

**Return:** the imputed version of $\mathbf{X}$.

For the setting of the proposed CBMI method, we use the default missForest setting for $\gamma$. That means, for a set continuous features $F$, $\gamma$ is governed by

$$\Delta = \frac{\sum_{j \in F} (\mathbf{X}_{new}^{imp} - \mathbf{X}_{old}^{imp})^2}{\sum_{j \in F} (\mathbf{X}_{new}^{imp})^2}, \tag{1}$$

while for a set of categorical features $G$, $\gamma$ is governed by

$$\Delta_G = \frac{\sum_{j \in G} \sum_{i=1}^{n} \mathbf{I}_{\mathbf{X}_{new}^{imp} \neq \mathbf{X}_{old}^{imp}}}{\#NA}. \tag{2}$$

Here, $\#NA$ is the number of missing entries in the categorical features.

## IV. Methodology

### A. Using labels for imputation

In this section, we detail the idea of ***imputation using labels (IUL)***. For clarity, from now, we refer to the common imputation procedure that only relies on the input itself as ***direct imputation (DI)***.

The IUL process is straightforward and is presented in algorithm 2. Suppose that we have input data $\mathbf{X}_{train}$ and labels $\mathbf{y}_{train}$. Then, the procedure starts by stacking the input training data $\mathbf{X}_{train}$ and labels $\mathbf{y}_{train}$ in a column-wise manner in step 1. Then, in step 2, the algorithm uses the imputation method $I$ to impute $\mathcal{D}_{train}$. This gives $\mathcal{D}^{imp}_{train}$. Then (step 3), $\mathbf{X}^{imp}_{train}$, the imputed version of $\mathbf{X}_{train}$ is the submatrix of $\mathcal{D}_{train}$ that consists of all columns except the last column of $\mathcal{D}_{train}$. In addition (step 4), $\mathbf{y}^{imp}_{train}$, the imputed version of $\mathbf{y}_{train}$ (if $\mathbf{y}_{train}$ contains missing values), is simply the last column of $\mathcal{D}^{imp}_{train}$. The algorithm ends by returning $\mathbf{X}^{imp}_{train}, \mathbf{y}^{imp}_{train}$.

It is important to note that the algorithm can handle not only $\mathbf{X}_{train}$ with missing data, but also $\mathbf{y}_{train}$ with missing data.

**Algorithm 2 IUL algorithm**

**Input:**

1) input training data $\mathbf{X}_{train}$, labels $\mathbf{y}_{train}$
2) imputation algorithm $I$.

**Procedure:**

1: $\mathcal{D}_{train} = [\mathbf{X}_{train}|\mathbf{y}_{train}]$
2: $\mathcal{D}^{imp}_{train} \leftarrow$ the imputed version of $\mathcal{D}_{train}$ using the imputation algorithm $I$.
3: $\mathbf{X}^{imp}_{train} \leftarrow$ imputed version of $\mathbf{X}_{train}$ that consists of all columns except the last column of $\mathcal{D}_{train}$ ,
4: $\mathbf{y}^{imp}_{train} \leftarrow$ imputed version of $\mathbf{y}_{train}$ if $\mathbf{y}_{train}$ contains missing values. This corresponds to the last column of $\mathcal{D}^{imp}_{train}$.

**Return:** $\mathbf{X}^{imp}_{train}, \mathbf{y}^{imp}_{train}$

### B. Classification via imputation

In this section, we detail the methodology of the proposed **Classification Based on MissForest Imputation (CBMI)** method. The algorithm is presented in algorithm 3.

Suppose that we have a dataset that consists of the training set $\{\mathcal{X}_{train}, \mathbf{y}_{train}\}$ that has $n_{train}$ samples and the testing set $\{\mathcal{X}_{test}, \mathbf{y}_{test}\}$ that has $n_{test}$ samples. Then, the procedure is as follows:

At the first step of the algorithm, we stack the training input $\mathcal{X}_{train}$ and the training label $\mathbf{y}_{train}$ in a column-wise manner. Next, in step 2, we initiate the predicted label of the test set $\hat{\mathbf{y}}$ with missing values, denoted by $*$. After that, in step 3, we stack the input of the test set $\mathcal{X}_{test}$ and $\hat{\mathbf{y}}$ in a column-wise manner, and we call this $\mathcal{D}^{*}_{test}$.

In the fourth step, we stack $\mathcal{D}_{train}$ and $\mathcal{D}^{*}_{test}$ in a row-wise manner. This gives us $\mathcal{D}^{*}$. Then, at step 5, we impute $\mathcal{D}^{*}$ using the missForest algorithm. This gives us an imputed matrix $\mathcal{D}$ that contains the imputed training, testing input, and the imputed label of the training set if the training data has missing labels and the imputed label of the test set.

**Algorithm 3 CBMI algorithm**

**Input:**

1) a dataset consists of the training set $\{\mathcal{X}_{train}, \mathbf{y}_{train}\}$ that has $n_{train}$ samples and the testing set $\{\mathcal{X}_{test}, \mathbf{y}_{test}\}$ that has $n_{test}$ samples,

**Procedure:**

1: $\mathcal{D}_{train} = [\mathcal{X}_{train}|\mathbf{y}_{train}]$
2: $\hat{\mathbf{y}} = \begin{pmatrix} * \\ \vdots \\ * \end{pmatrix}$: an NA-initiated vector of size $n_{test}$ where $n_{test}$ is the number of sample in the test set
3: $\mathcal{D}^{*}_{test} = [\mathcal{X}_{test}|\hat{\mathbf{y}}]$
4: $\mathcal{D}^{*} = \begin{pmatrix} \mathcal{D}_{train} \\ \mathcal{D}^{*}_{test} \end{pmatrix}$
5: $\mathcal{D} \leftarrow$ the imputed version of $\mathcal{D}^{*}$ using the missForest algorithm

**Return:** $\mathcal{D}$, the imputed version of $\hat{\mathbf{y}}$ obtained from $\mathcal{D}$.

## V. Experiments

### A. Experiment set up

We compare imputation using labels (IUL) with the direct imputation on training data approach (DI) by splitting a dataset into training and testing sets with ratio 6:4 and simulating missing rates $20\% - 80\%$ on the training input. Here, the missing rate is defined as the ratio between the number of missing entries and the total number of entries. Each experiment is repeated 10 times, and we plot the mean accuracy. We use MissForest as the imputation method with default configuration and Random Forest as the classifier.

In addition, we compare CBMI with the two-step approach of *Imputation* and then *Classification* using Random Forest Classifier [20], abbreviated as IClf. The reason for choosing Random Forest as the Classifier is to facilitate fair comparison with CBMI, which relies on Random Forest to impute missing values. The stopping criterion for missForest is the default in the missingpy package [1].

The datasets for the experiments are from the UCI Machine Learning repository [22], and are described in table I. For each dataset, we simulate missing data with missing rates $r$ ranging between 0% - 80%. Each experiment is repeated 10 times and the training to testing ratio is 6:4, and we report the mean $\pm$ standard deviation from the runs for both the accuracy and the running time. We also conducted the experiments in two settings: the test set is fully observed and the test set has missing values. In case the test set is fully observed, the missing rate $r$ is the missing rate for simulating missing values on training data only. In the case where there are missing values in the dataset, the missing values are generated with the same missing rates as in the training input.

[1] https://pypi.org/project/missingpy/

TABLE I
DATASETS FOR EXPERIMENTS

| Datasets | # samples | # features |
|---|---|---|
| liver | 345 | 6 |
| soybean | 47 | 35 |
| Parkinson | 195 | 22 |
| heart | 267 | 44 |

For a fair comparison, for IClf, if the test set contains missing data then it is merged with train set for imputation. Note that this is only for the imputation of test set only. The imputation of the training data is independent of the testing data.

### B. Results and Analysis for IUL

The results for the experiments on the performance of IUL are shown in figures 1, 2, 3, 4. From the plots, we can see that IUL consistently outperforms the DI approach of imputation that only relies on the input, especially at the high missing rates.

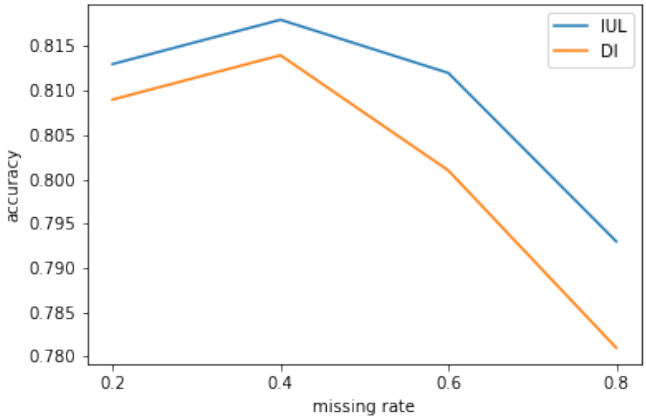


Fig. 1. Results on the heart dataset.

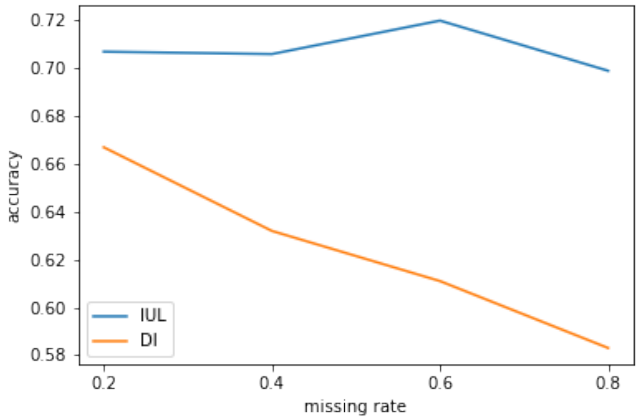


Fig. 2. Results on the liver dataset.

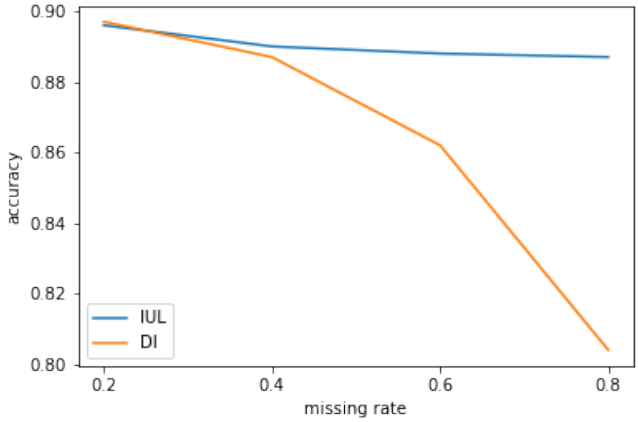


Fig. 3. Results on the parkinson dataset.

### C. Results and Analysis for CBMI

From the tables of results, we see that most of the time, CBMI achieves better results than IClf. For example, at missing rate $r = 80\%$, for the soybean dataset when missing data present in the test set (table III), the accuracy of CBMI is 0.547, while it is 0.5 for IClf. This implies an improvement of 4.7 % when using CBMI compared to IClf. In addition, most of the time, the standard deviation of the accuracy obtained from CBMI is also smaller than IClf.

Interestingly, CBMI achieves better results than IClf for many cases where the missing rate $r$ is 0, i.e., there is no missing data in the training set. For example, when the training data is fully observed ($r = 0$) on the heart dataset with missing data present in the test set (table V), CBMI gives an accuracy of 0.821, while IClf gives 0.817; or on the Parkinson dataset where the test set is fully observed (table VIII), CBMI's accuracy is 0.82, while IClf's accuracy is 0.818.

Regarding the running time, when missing data present in the test set, from tables II, III, IV and V CBMI is clearly faster. However, from tables VI, VII, VIII, IX when the testing data is fully observed, IClf is slightly faster than CBMI most of the time. This is possibly because when the testing data is fully observed, CBMI's imputation process has to impute a larger matrix, which consists of the training and testing input and labels, while IClf imputes only the training input.

## VI. CONCLUSION AND FUTURE WORKS

In this work, we have illustrated how using the training labels for imputation can significantly improve the imputation of training input. In addition, we have introduced CBMI, a classification method for missing data that relies on missForest imputation to impute the label of testing data. The algorithm returns not only the label of the testing data but also the

TABLE II
CLASSIFICATION RESULTS ON THE LIVER DATASET WHEN MISSING DATA PRESENTS IN THE TEST SET

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.7±0.031 | 0.704±0.031 | 0.392±0.019 | 0.102±0.007 |
| 20 % | 0.649±0.022 | 0.642±0.019 | 5.379±1.778 | 7.094±1.237 |
| 40 % | 0.587±0.046 | 0.583±0.028 | 5.658±1.421 | 7.662±1.659 |
| 60 % | 0.541±0.041 | 0.562±0.042 | 5.715±1.594 | 7.502±1.428 |
| 80 % | 0.554±0.025 | 0.547±0.048 | 5.704±1.685 | 7.774±1.775 |

TABLE III
CLASSIFICATION RESULTS ON THE SOYBEAN DATASET WHEN MISSING DATA PRESENTS IN THE TEST SET

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.995±0.016 | 0.995±0.016 | 0.352±0.012 | 0.089±0.008 |
| 20 % | 0.989±0.021 | 0.984±0.034 | 24.579±5.383 | 45.279±6.573 |
| 40 % | 0.916±0.116 | 0.9±0.106 | 27.251±6.12 | 44.467±5.554 |
| 60 % | 0.821±0.092 | 0.8±0.091 | 29.487±5.299 | 44.25±9.486 |
| 80 % | 0.547±0.142 | 0.5±0.127 | 26.82±6.247 | 48.91±8.085 |

TABLE IV
CLASSIFICATION RESULTS ON THE PARKINSON DATASET WHEN MISSING DATA PRESENTS IN THE TEST SET

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.896±0.027 | 0.903±0.032 | 0.368±0.006 | 0.092±0.003 |
| 20 % | 0.877±0.024 | 0.869±0.038 | 15.456±4.69 | 27.058±5.667 |
| 40 % | 0.862±0.045 | 0.858±0.043 | 18.927±3.824 | 26.745±2.528 |
| 60 % | 0.795±0.035 | 0.813±0.039 | 18.949±4.077 | 22.418±5.805 |
| 80 % | 0.778±0.037 | 0.769±0.06 | 18.711±4.516 | 27.507±5.776 |

TABLE V
CLASSIFICATION RESULTS ON THE HEART DATASET WHEN MISSING DATA PRESENTS IN THE TEST SET

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.821±0.031 | 0.817±0.021 | 0.363±0.008 | 0.098±0.004 |
| 20 % | 0.83±0.038 | 0.83±0.036 | 42.814±8.138 | 74.018±12.008 |
| 40 % | 0.809±0.048 | 0.813±0.037 | 50.46±5.902 | 87.255±11.24 |
| 60 % | 0.806±0.025 | 0.801±0.029 | 47.827±5.954 | 86.801±9.228 |
| 80 % | 0.779±0.036 | 0.783±0.05 | 48.81±4.614 | 90.821±7.51 |

TABLE VI
CLASSIFICATION RESULTS ON THE LIVER DATASET WHEN THE TEST INPUT IS FULLY OBSERVED

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.707±0.029 | 0.712±0.035 | 0.395±0.002 | 0.095±0.002 |
| 20 % | 0.687±0.032 | 0.67±0.015 | 5.898±1.426 | 3.664±0.886 |
| 40 % | 0.664±0.037 | 0.624±0.037 | 5.209±2.157 | 4.223±0.565 |
| 60 % | 0.646±0.056 | 0.632±0.049 | 6.262±1.813 | 4.115±1.322 |
| 80 % | 0.568±0.033 | 0.565±0.046 | 6.162±1.465 | 2.558±1.065 |

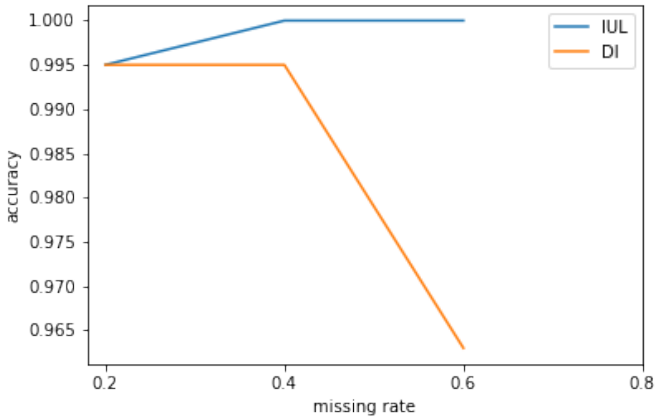


Fig. 4. Results on the soybean dataset.

TABLE VII
CLASSIFICATION RESULTS ON THE SOYBEAN DATASET WHEN THE TEST INPUT IS FULLY OBSERVED. MISSING RATE 80% GIVES SOME SAMPLES THAT ARE COMPLETELY EMPTY. SO WE DISCARD THIS CASE.

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.995±0.016 | 1.0±0.0 | 0.709±0.052 | 0.116±0.014 |
| 20 % | 0.995±0.016 | 0.995±0.016 | 57.483±14.496 | 63.033±8.178 |
| 40 % | 0.989±0.021 | 0.963±0.062 | 50.181±15.803 | 43.733±9.249 |
| 60 % | 0.995±0.016 | 0.937±0.061 | 27.561±11.688 | 25.348±9.366 |

imputed version of the training input and the testing input. CBMI is also capable of handling input with missing labels. In the future, we want to further explore the capability of CBMI for handling input with missing labels, and application in semi-supervised learning.

TABLE VIII
CLASSIFICATION RESULTS ON THE PARKINSON DATASET WHEN THE TEST INPUT IS FULLY OBSERVED

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.904±0.037 | 0.895±0.045 | 0.368±0.006 | 0.092±0.003 |
| 20 % | 0.863±0.041 | 0.871±0.05 | 16.72±3.905 | 15.741±4.267 |
| 40 % | 0.865±0.026 | 0.862±0.033 | 18.888±4.622 | 13.52±1.636 |
| 60 % | 0.873±0.032 | 0.847±0.045 | 16.582±5.786 | 10.715±3.629 |
| 80 % | 0.79±0.071 | 0.799±0.05 | 19.093±5.147 | 13.792±5.38 |

TABLE IX
CLASSIFICATION RESULTS ON THE HEART DATASET WHEN THE TEST INPUT IS FULLY OBSERVED

| | accuracy | | running time | |
|---|---|---|---|---|
| $r$ | CBMI | IClf | CBMI | IClf |
| 0 % | 0.82±0.017 | 0.818±0.023 | 0.375±0.004 | 0.101±0.001 |
| 20 % | 0.813±0.04 | 0.808±0.039 | 38.631±8.816 | 35.641±5.884 |
| 40 % | 0.819±0.031 | 0.815±0.039 | 46.706±9.855 | 41.273±6.889 |
| 60 % | 0.825±0.032 | 0.811±0.032 | 44.298±8.777 | 42.332±8.138 |
| 80 % | 0.777±0.04 | 0.78±0.038 | 45.646±7.224 | 37.522±7.628 |

## REFERENCES


[1] M. A. Vu, T. Nguyen, T. T. Do, N. Phan, P. Halvorsen, M. A. Riegler, and B. T. Nguyen, "Conditional expectation for missing data imputation," *arXiv preprint arXiv:2302.00911*, 2023.

[2] D. J. Stekhoven and P. Bühlmann, "Missforest-non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2012.

[3] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[4] M. M. Ghazi, M. Nielsen, A. Pai, M. J. Cardoso, M. Modat, S. Ourselin, and L. Sørensen, "Robust training of recurrent neural networks to handle missing data for disease progression modeling," *arXiv preprint arXiv:1808.05500*, 2018.

[5] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, pp. 1–68, 2010.

[6] L. Gondara and K. Wang, "Multiple imputation using deep denoising autoencoders," *arXiv preprint arXiv:1705.02737*, 2017.

[7] S. M Mostafa, A. S Eladimy, S. Hamad, and H. Amano, "Cbrl and cbrc: Novel algorithms for improving missing value imputation accuracy based on bayesian ridge regression," *Symmetry*, vol. 12, no. 10, p. 1594, 2020.

[8] P. Keerin, W. Kurutach, and T. Boongoen, "An improvement of missing value imputation in dna microarray data using cluster-based lls method," in *2013 13th International Symposium on Communications and Information Technologies (ISCIT)*. IEEE, 2013, pp. 559–564.

[9] S. J. Choudhury and N. R. Pal, "Imputation of missing data with neural networks for classification," *Knowledge-Based Systems*, vol. 182, p. 104838, 2019.

[10] A. Garg, D. Naryani, G. Aggarwal, and S. Aggarwal, "Dl-gsa: a deep learning metaheuristic approach to missing data imputation," in *International Conference on Sensing and Imaging*. Springer, (2018), pp. 513–521.

[11] K. Mohan and J. Pearl, "Graphical models for processing missing data," *Journal of the American Statistical Association*, pp. 1–42, 2021.

[12] J. Fan, Y. Zhang, and M. Udell, "Polynomial matrix completion for missing data imputation and transductive learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, (2020), pp. 3842–3849.

[13] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh, "Matrix completion and low-rank svd via fast alternating least squares," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 3367–3402, 2015.

[14] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

[15] M. G. Rahman and M. Z. Islam, "Missing value imputation using a fuzzy clustering-based em approach," *Knowledge and Information Systems*, vol. 46, no. 2, pp. 389–422, 2016.

[16] S. I. Khan and A. S. M. L. Hoque, "Sice: an improved missing data imputation technique," *Journal of big data*, vol. 7, no. 1, pp. 1–21, 2020.

[17] J. S. Murray and J. P. Reiter, "Multiple imputation of missing categorical and continuous values via bayesian mixture models with local dependence," *Journal of the American Statistical Association*, vol. 111, no. 516, pp. 1466–1479, 2016.

[18] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[21] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006.

[22] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml