Cao YH, Wu JX. Random Subspace Sampling for Classification with Missing Data. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(1): 1–19 January 2018. DOI 10.1007/s11390-015-0000-0

Random Subspace Sampling for Classification with Missing Data

Yun-Hao Cao[1] (曹云浩), and Jian-Xin Wu[1,*] (吴建鑫), *Member, CCF, IEEE*

[1] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*

E-mail: caoyh@lamda.nju.edu.cn; wujx2001@nju.edu.cn

Received May 26, 2021; accepted November 3, 2022.

Abstract Many real-world datasets suffer from the unavoidable issue of missing values, and therefore classification with missing data has to be carefully handled since inadequate treatment of missing values will cause large errors. In this paper, we propose a random subspace sampling (RSS) method by sampling missing items from the corresponding feature histogram distributions in random subspaces, which is effective and efficient at different levels of missing data. Unlike most established approaches, RSS does not train on fixed imputed datasets. Instead, we design a dynamic training strategy where the filled values change dynamically by resampling during training. Moreover, thanks to the sampling strategy, we design an ensemble testing strategy where we combine the results of multiple runs of a single model, which is more efficient and resource-saving than previous ensemble methods. Finally, we combine these two strategies with the random subspace method, which makes our estimations more robust and accurate. The effectiveness of the proposed method is well validated by experimental studies.

Keywords missing data, random subspace, neural networks, ensemble learning

1 Introduction

Classification is one of the most important tasks in machine learning and data mining. Many algorithms have been proposed to deal with classification problems, but the majority of them require complete data and cannot be directly applied to data with missing values. Even for algorithms that can cope with incomplete data, missing values can often result in large classification errors[1]. Unfortunately, missing values are a common issue in numerous real-world applications. For example, 45% of the datasets in the UCI machine learning repository[1], which is one of the most popular benchmark databases, contain missing values.

The simplest approach for dealing with missing values is to ignore those instances with missing attributes. Commonly referred to as the removal approaches, such techniques are clearly suboptimal when a large portion of the data has missing attributes, and of course infeasible, if each instance is missing at least one or more features. A more pragmatic approach commonly used to accommodate missing data is to use imputation methods to substitute missing values with plausible values. For example, mean imputation replaces all missing values in a feature with the average of existing values in the same feature. Imputation can provide complete data which can then be used by any classification algorithm. Single-imputation methods such as mean imputation are often efficient but they are not accurate enough. In contrast, multiple-imputation methods such as [2] create multiple imputed datasets to reflect better the uncertainty in incomplete data. They are usually more accurate but are

Regular Paper
This work was supported by the National Natural Science Foundation of China under Grant No. 61772256 and No. 61921006.
*Corresponding Author

[1] http://archive.ics.uci.edu/ml Oct. 2022

computationally expensive[3]. It remains a challenge to determine how to combine classification algorithms and imputation in a way that is both effective and efficient.

With the rapid development of ensemble learning, there have also been ensemble methods for classification with missing data[4]. For example, Krause and Polikar[5] trained an ensemble of base classifiers with random subsets of features to classify with missing data. These methods build multiple classifiers in the training process and then applicable classifiers are selected to classify each incomplete instance during inference. However, existing ensemble methods for classification with missing data often cannot work well on datasets with numerous missing values[6]. More importantly, they cannot guarantee to classify all incomplete instances. Hence, how to develop an ensemble method that is effective even when the data contains many missing values is still a challenge.

In this paper, we take a step towards designing an efficient and effective method at different levels of missing data, by combining the advantages of multiple-imputation and ensemble learning. We propose a Random Subspace Sampling (RSS) method for classification with missing data, which first constructs different random subspaces and corresponding base learners. Then, for each missing item in each random subspace, we directly sample from the corresponding feature histogram distribution to fill in. During the training stage, we design a dynamic training strategy where we resample and probabilisitically change the filled value for each missing item. During the inference stage, thanks to our sampling strategy, we design an ensemble testing strategy where we combine the results of multiple runs of a single model, which is efficient and effective. In contrast to multiple-imputation methods which need iterative steps to impute, ours is more efficient by sampling directly. Moreover, the dynamic training strategy distinguishes our method from most established approaches that train on fixed data after imputation.

Experimental results validate the effectiveness of RSS. We achieve superior performance on six incomplete datasets with inherent missing values and nine complete datasets at four levels of artificially introduced missing values. Furthermore, we carefully study the impact of each component in RSS through ablation studies and the sensitivity of hyperparameters.

2 Related Work

This section discusses related work, including traditional and ensemble methods for classification with missing data.

2.1 Traditional Methods for Missing Data

There are four major approaches to addressing classification with missing data: the removal approach, the model-based approach, the machine learning approach and the imputation approach[1]. The removal approach simply deletes all instances containing missing values, which is limited to datasets with only a few missing values in the training data and no missing values during inference. The model-based approach generates a data distribution model from input data. One of the most used approaches in this category is the mixture models trained with the expectation-maximization (EM) algorithm. Zoubin and Michael[7] trained Gaussian Mixture Models (GMM) on incomplete data using the EM algorithm. Ghahramani and Jordan[8] proposed Bayesian techniques for estimating class probabilities from incomplete data using neural networks. Although this approach can classify both complete and incomplete instances, it requires making assumptions about the joint distribution of all features in the model[1]. The machine learning approach makes classifiers that are able to directly classify incomplete datasets, e.g., C4.5[9].

However, this approach usually suffers from limited classification accuracies.

The most used approach to classification with incomplete data is to use imputation methods to transform incomplete data into complete data before building a classifier in the training process or classifying a new incomplete instance in the application process. This approach has the advantage that the imputed complete data can be used by any classification algorithm. Single-imputation methods such as mean and KNN imputation[10] provide a simple missing data imputation but under-represent the variability in the data[11]. Zhao and Udell[12] developed an approximate EM algorithm to estimate copula parameters from incomplete mixed data. Instead of filling in a single value for each missing one, multiple-imputation methods[2, 13, 14, 15] impute the missing values for H times to produce H complete datasets using an appropriate model that incorporates random variation. Multiple-imputation methods have become more and more popular because they reflect better uncertainty and often yield better performance. However, they are computationally very expensive. Despite sharing the similarity that our method also uses multiple plausible values for each missing item to reflect better uncertainty, our method differs from multiple-imputation methods in at least three aspects: 1) We directly sample from the estimated histogram distribution for missing values, which is far more efficient than the iterative steps in multiple-imputation methods. (2) By using the ensemble testing strategy, we can ensemble multiple predictions without the need to generate multiple datasets and train multiple models. 3) We adopt a novel dynamic training strategy where the imputed values dynamically change during training and it distinguishes our method from previous approaches.

2.2 Ensemble Methods for Missing Data

Ensemble learning is a powerful learning paradigm which constructs a set of base classifiers for classification and it has become the choice for many industrial applications and data science projects[16]. The random subspace method (RSM), which was originally proposed by Ho[17], is the pillar of many ensemble methods, e.g., random forests[18]. In RSM, classifiers are trained using different random subsets of the features, allowing classifiers to err in different sub-domains of the feature space.

Ensemble methods, especially RSM, have also been used for classification with missing data. One of the earliest studies using ensembles for classification appeared in [19], where four neural networks are built to address classification with a thyroid database consisting of two incomplete features. Stefan and Robi[5] trained an ensemble of base classifiers with random subsets of features to classify with missing data. In [20], the incomplete dataset is divided into a group of complete sub-datasets, which are then used as the training sets for neural networks. In these approaches, when an incomplete instance needs to be classified, only those classifiers trained with those features that are available in the instance are used to classify the instance. Although these methods can cope with incomplete data to some extent, they usually do not obtain good accuracy when datasets contain a large number of missing values. The underlying reason is that the complete sub-datasets often only have a small number of instances for base classifiers to train on when the datasets include a large number of missing values. Moreover, they cannot guarantee to classify all incomplete instances, especially when data contains many missing values. In contrast, our method differs as follows: 1) We develop our method from NRS[21], which implements RSM in the context of

neural networks and enables our method to enjoy the benefits of both ensemble learning and representation learning. 2) All instances are used in the training of each base classifier, which ensures that we make full use of all information. 3) Our method can classify all incomplete instances well even when the dataset contains many missing values.

3 Random Subspace Sampling

In this section, we propose the random subspace sampling (RSS) method. Recently, Cao *et al.*[21] proposed a neural random subspace (NRS) method, which implements the random subspace idea in the context of neural networks and has achieved impressive results on various tasks. In this paper, we develop our method based on NRS. First, we introduce the notation used in this paper. We then revisit the NRS method and introduce our RSS method.

Let $D = \{(\boldsymbol{x}^{(i)}, y^{(i)})|i = 1, \cdots, n\}$ denote a dataset, where each $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ represents an input instance with its associated label $y^{(i)} \in \{1, \cdots, K\}$, n is the number of instances, d is the number of features and K is the number of classes. Each instance $\boldsymbol{x}^{(i)}$ is represented by a d-dimensional feature vector $(x_1^{(i)}, x_2^{(i)}, \cdots, x_d^{(i)})$, where an $x_j^{(i)}$ is either a valid value of the j-th feature, or is the value "?", which means that its value is unknown (a missing value).

3.1 NRS Recap

NRS has three hyper-parameters, namely, the depth expansion rate $nMul$, the height/width expansion rate dH/dW and the number of channels per group in the group convolution $nPer$. For a d-dimensional feature vector $\boldsymbol{x}$, NRS first generates $q = dH \times dW \times nMul$ randomly permuted vectors from $\boldsymbol{x}$, each of which is d-dimensional. Then, these q feature vectors are arranged into an order-3 tensor $\boldsymbol{X} \in \mathbb{R}^{dH \times dW \times C}$, where $C = nMul \times d$. $\boldsymbol{X}$ includes a set of 2D feature maps $\boldsymbol{X} = \{\boldsymbol{X}^c|c = 1, \cdots, C\}$. Each feature map $\boldsymbol{X}^c$ consists of $g = dH \times dW$ features, which are randomly selected from the original features, that is, it is a random subspace. Hence, C feature maps correspond to C random subspaces. Then, a depthwise group convolution plus the subsequent ReLU non-linearity is acted upon $\boldsymbol{X}$ to get $\boldsymbol{S} = (S_1, \cdots, S_C)$, where each S_c represents the output for the c-th random subspace $\boldsymbol{X}^c$:

$$S_c = f\left(\sum_i \sum_j \boldsymbol{X}^c(i,j)\boldsymbol{W}^c(i,j)\right), \tag{1}$$

where $f(\cdot)$ denotes the ReLU function and each $\boldsymbol{W}^c$ $(c = 1, \cdots, C)$ denotes the weights of the c-th depthwise convolution filter. Finally, a fully connected layer plus a softmax layer are used to combine all base classifiers' outputs $\{S_c|c = 1, \cdots, C\}$ for classification:

$$Y_k = \sum_{c=1}^{C} S_c \boldsymbol{w}^{\mathrm{FC}}(c,k), \tag{2}$$

where $\boldsymbol{w}^{\mathrm{FC}}$ denotes the weight matrix (of size $C \times K$) of the fully connected layer and Y_k denotes the output for the k-th class $(k = 1, \cdots, K)$. We set $dH/dW = 3$ and $nPer = 1$ as done in [21] in this paper.

In short, NRS implements the random subspace idea in neural networks both efficiently and effectively. Hence, we develop our algorithm on the basis of it and will further study the impact of the selected NRS architecture.

3.2 The Proposed Method

Now, we introduce our random subspace sampling algorithm, which mainly contains three steps:

1) *Estimating Histogram Distribution.* First, we estimate the histogram distribution individually at each feature dimension, which will be introduced next.

2) *Constructing Random Subspaces.* Then, we construct random subspaces following NRS.

3) *Sampling for Missing Features.* To handle the missing data problem, we sample from the corresponding histogram distribution for each missing item individually in each random subspace and substitute it with the sampled value.

Histogram Distribution. We calculate the histogram distribution for each feature individually in the train set and we disregard all missing values at this feature dimension when counting. Each feature x_j can be either a categorical or a continuous variable and histogram can handle both situations well. Next we consider the continuous situation and assume that $x_j \in [a, b]$, so $p_j(x)$ is non-zero only within $[a, b]$. The histogram is to partition the set $[a, b]$ into several bins and uses the count of features falling into the bin as a density estimate. When we have M bins, this yields a partition:

$$B_1 = [a, a + \frac{b-a}{M}), \cdots, B_M = [a + \frac{(M-1)(b-a)}{M}, b] .$$

Then, for a given point $x \in B_l$, the density estimate from the histogram will be

$$\hat{p}_j(x) = \frac{M}{n(b-a)} \sum_{i=1}^{n} I(x_j^{(i)} \in B_l) ,$$

where $I(\cdot)$ is the indicator function. Note that a missing value does not belong to any bins. Then, for any given point x, the probability of $x \in B_l$ is

$$P(x \in B_l) = \int_{B_l} \hat{p}_j(x) \mathrm{d}x = \frac{1}{n} \sum_{i=1}^{n} I(x_j^{(i)} \in B_l) .$$

We use the average value of the endpoints in B_l as a representative, hence we define the histogram distribution of the j-th feature ($j = 1, 2, \cdots, d$) as

$$P_j^{\text{hist}}(x = a + \frac{(2l-1)(b-a)}{2M}) = \frac{1}{n} \sum_{i=1}^{n} I(x_j^{(i)} \in B_l) , \tag{3}$$

where $l = 1, 2, \cdots, M$.

Proposition 1. *P_j^{hist} is a valid probability distribution.*

Proof. $\int_x p_j^{\text{hist}}(x)\mathrm{d}x = \frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{M} I(x_j^{(i)} \in B_l) = 1$ and obviously we have $P_j^{\text{hist}}(x) \geq 0$. □

We use the histogram distribution $P_j^{\text{hist}}(x)$ ((3)) to sample the j-th feature if it is missing in the subsequent training and inference processes. Histogram visualizations on several datasets used in this paper will be shown in Subsection 4.1.1.

Random Subspace Sampling. Then, we construct random subspaces following the steps in NRS[21], as shown in Fig. 1. We have C random subspaces $\boldsymbol{X}^1, \cdots, \boldsymbol{X}^C$ in total, where each $\boldsymbol{X}^c$ ($c = 1, \cdots, C$) contains g features (C and g are defined as before). Then for each feature $x_j \in \boldsymbol{X}^c$, if it is missing then we sample from the corresponding histogram distribution P_j^{hist} and substitute the missing item with the sampled value. Notice that one feature will appear multiple times in all random subspaces and we independently resample for each missing item to impute every time it appears (thus can have different sampled values). The pseudo code of RSS is shown in Algorithm 1.

During training, we design a dynamic training strategy where we resample all missing items at the start of each epoch. In other words, we are changing filled values for missing items dynamically, which distinguishes our algorithm from other imputation-based methods.

During inference, notice that our algorithm will generate different outputs if we run it multiple times for the same test instance, which is due to the random sampling method for missing values. More specifically, we resample the missing values and run the model for H times and each time we generate $\boldsymbol{Y} = (Y_1, \cdots, Y_K)$ according to (2), as shown in Algorithm 1. However, this is not a disadvantage but a benefit, because we can run the single model multiple times and ensemble these predictions. Compared to other ensemble techniques which require training multiple models to ensemble,

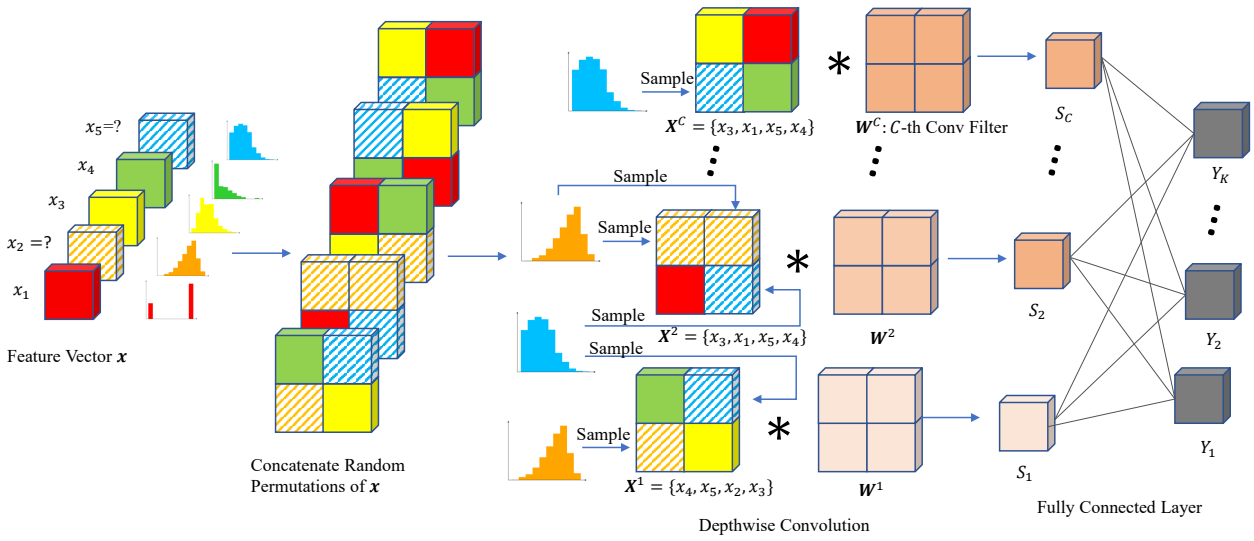


Fig.1. RSS architecture. The input feature vector is $\boldsymbol{x} = (x_1, \cdots, x_5)$, where x_2 and x_5 are missing and marked with diagonal lines. For better illustration, we set $nMul$ to 1 and hence $C = 5$ and $\boldsymbol{X} = \{\boldsymbol{X}^1, \cdots, \boldsymbol{X}^5\}$.

e.g., multiple-imputation methods, ours uses a single model by running it multiple times, which is far more resource-efficient. We will further study the impact of our training and inference strategy in ablation studies.

Notice that the training and inference strategies above are not limited to NRS and can also be applied to other architectures, e.g., multi-layer perceptrons (MLP). In ablation studies, we show that 1) the two strategies are effective in both NRS and MLP, 2) the inherent random subspace method is crucial in our RSS since it produces larger improvements than in MLP and consequently the highest accuracies when combined with the above two strategies.

3.3 Analysis about Feature Interactions

Notice that when filling in the missing values using histogram random sampling, we are only considering individual features without considering the interactions between features. Most of the time the randomly generated values are not applicable to the sample and now we discuss how our method inherently utilizes feature interactions and filter out inappropriate values. Here we use the example in Fig. 1 and specifically, we use the last subspace $\boldsymbol{X}^C$ for illustration. As shown in Fig. 1, $\boldsymbol{X}^C$ includes 1 missing feature x_5 and 3 known features x_1, x_3 and x_4. Hence, according to (1), we have:

$$S_C = f\left(W_1^C x_3 + W_2^C x_1 + W_3^C x_5 + W_4^C x_4\right),$$

where we abbreviate $\boldsymbol{W}^C(i,j)$ as W^C_{2i+j} in this case. Without loss of generality, we assume $W_3^C > 0$. We can notice that S_C is activated after the ReLU function $f(\cdot)$ if and only if

$$x_5 > -\frac{1}{W_3^C}(W_1^C x_3 + W_2^C x_1 + W_4^C x_4),$$

where the "$>$" becomes "$<$" if $W_3^C < 0$. In other words, we inherently utilize interactions between missing feature x_5 and known features x_1, x_3 and x_4 by a linear combination and the non-linearity in f.

We update these weights $\boldsymbol{W}^C$ during training and inappropriate fill-in values x_5 will deactivate S_C, i.e., $S_C = 0$. With more such random subspaces, adjustable weights and non-linearities, we can further explore the interactions between features. During inference, we can filter out incorrect fill-in values through multiple sampling, thanks to such feature interactions.

Algorithm 1 Random Subspace Sampling

Input: training dataset $D_{tr} = \{(\boldsymbol{x}^{(i)}, y^{(i)})|i = 1, \cdots, n\}$, test dataset $D_{te} = \{(\boldsymbol{x}^{(i)}|i = n+1, \cdots, n+m\}$, network $f(\cdot;\theta)$, total number of bins M, total training epochs T, test ensemble size H;
Output: predicted labels $\hat{Y}_{te} = \{\hat{y}^{(i)}|i = n+1, \cdots, n+m\}$ on the test dataset;

Estimate histogram distributions P_j^{hist} $(j = 1, \cdots, d)$ on D_{tr} using M bins in (3);
▷ **Training process:**
for $t = 1, \cdots, T$ **do**
  for $i = 1, \cdots, n$ **do**
    Concatenate and reshape random permutations of $\boldsymbol{x}^{(i)}$ and get 2D feature maps $\{\boldsymbol{X}^c|c = 1, \cdots, C\}$;
    for $c = 1, \cdots, C$ **do**
      if $x_j^{(i)} \in \boldsymbol{X}^c$ and $x_j^{(i)} ==$"?" **then**
        Sample from P_j^{hist} and substitute the missing position in $\boldsymbol{X}^c$ with the sampled value;
      end if
    end for
    Train the network $f(\cdot;\theta)$ as normal;
  end for
end for
▷ **Inference process:**
for $i = n+1, \cdots, n+m$ **do**
  for $h = 1, \cdots, H$ **do**
    Sample for missing items as before and get the network output $\boldsymbol{Y}^{(i,h)} = (Y_1^{(i,h)}, \cdots, Y_K^{(i,h)}) = f(\boldsymbol{x}^{(i)};\theta)$ using (2), where the superscript (i,h) denotes the output for the i-th instance in the h-th iteration;
  end for
  Ensemble predictions for the i-th instance: $\hat{y}^{(i)} = \arg\max_k \sum_{h=1}^{H} Y_k^{(i,h)}$;
end for
return $\hat{Y}_{te} = \{\hat{y}^{(i)}|i = n, \cdots, n+m\}$.

3.4 Analysis about Time Complexity

Now we analyze the time complexity of our method as well as other comparison methods. As mentioned before, the dataset contains n training instances and m test instances, where each instance contains d features. We train all networks for T epochs. Notice that the total training time t_{train} contains two parts, i.e., the pre-processing time t_{pre} and the network training time t_{net}. Also, we denote the total test time as t_{test}.

Mean Imputation. Now the imputation time for each missing feature is $O(n)$ (calculating mean value) and the total pre-processing time is $t_{\text{pre}} = O(nd)$. The network training time is $t_{\text{net}} = O(nT)$ and $t_{\text{train}} = t_{\text{pre}} + t_{\text{net}} = O(nd + nT)$. The test time is $t_{\text{test}} = O(m)$.

KNN Imputation. For the KNN imputation, the imputation time for each instance is $O(nd)$ (calculating distance for one instance) and the total pre-processing time is $t_{\text{pre}} = O(n^2 d)$ (calculating distance matrix). Hence, the total training time is $t_{\text{train}} = O(n^2 d + nT)$. During the inference stage, we need to calculate the distance from the entire training set for each instance, hence the total test time is $t_{\text{test}} = O(mnd)$.

MICE Imputation. Each feature is modeled as a regression function of other features. Hence for each feature, the pre-processing time is $O(n^2 d)$ (solving the regression) and the total time is $t_{\text{pre}} = O(n^2 d^2)$. Hence, the total training time is $t_{\text{train}} = O(n^2 d^2 + nT)$. During the inference stage, each feature is calculated using other features, hence the test time is $O(md^2)$.

RSS. We only need to calculate the histogram distribution for each feature during the pre-processing, hence $t_{\text{pre}} = O(nd)$. We need to sample each missing feature during the training and hence our training cost is $t_{\text{net}} = O(nT + dnT)$. Hence, the total training time

is $t_{\text{train}} = O(ndT)$. We also need sampling during the inference stage and the test time is $t_{\text{test}} = O(md)$.

We will also empirically compare the running speed of each method in Subsection 4.6.

4 Experimental Results

In this section, we experimentally investigate the proposed method. First, we introduce the experimental settings and then we evaluate our method on 15 datasets. Then, we conduct ablation studies to investigate the impact of each component in RSS and also conduct experiments to study the sensitivity of hyperparameters in RSS. Finally, we carefully compare the performance of RSS with various combinations of different classifiers and imputation methods and also the running speed of RSS with other imputation methods.

4.1 Experimental Settings

4.1.1 Datasets

Fifteen datasets, summarized in Table 1, are used in the experiments. These are taken from the UCI repository of Machine Learning Databases[2]. Each dataset is presented in one row in Table 1, including the number of instances, the number of features, the number of classes and the proportion of missing values (PMV). The first six datasets suffer from missing values in a "natural" way. In these datasets, we do not know any information related to the randomness of missing values, so we assume that missing values in these datasets are distributed in a missing at random (MAR) way[22]. Fig. 2 shows the histogram distributions of the first five features on chess, letter, pendigits and segment. As can be seen, different features have different distributions (may be Gaussian, Uniform, bimodal, etc.) and histograms are a suitable way to describe these various distributions.

Table 1. Dataset Statistics and Hyper-parameter Settings

Datasets	Statistics				Settings
	#Instances	#Dim	#Classes	PMV (%)	$nMul$
Mammographics	961	5	2	3.37	100
Hepatitis	155	19	2	5.67	100
Kidney-disease	400	24	2	10.54	50
Horse	368	22	2	23.8	50
Pima	768	8	2	12.24	100
Bands	539	19	2	5.38	100
Dna	3186	180	3	0	5
Protein	24387	357	3	0	2
Chess-krkp	3196	36	2	0	20
Chess-krkopt	28056	6	18	0	50
Letter	20000	16	10	0	100
HTRU2	17898	9	2	0	100
Yeast	1484	8	14	0	100
Segment	2310	19	7	0	50
Pendigits	10992	10	16	0	50

Note: '#Instances' denotes the number of instances in the dataset, '#Dim' denotes the number of feature dimensions, '#Classes' denotes the number of classes.

In order to test the performance of the proposed method with datasets containing different levels of missing values, the missing completely at random (MCAR) mechanism is utilized to introduce missing values into the last nine complete datasets. Four different levels of missing values: 20%, 40%, 60% and 70% are used to introduce missing values into the datasets. For each complete dataset and each level of the four missing levels, we randomly separate the set into two subsets, one with 70% examples for training, and the other one with 30% examples for testing. We repeat the random partition 10 times and report the average results.

Considering the small number of samples in the first six incomplete datasets, we run 10 times in each partition and therefore 100 (=10×10) results are obtained for each of these datasets. For the last seven complete datasets, we report both the full test accuracy (where we use the complete test set) and missing test accuracy (where we use artificially created missing test set as in training). We only report missing test accuracy in ablation studies and hyperparameter studies.

[2]http://archive.ics.uci.edu/ml Oct. 2022

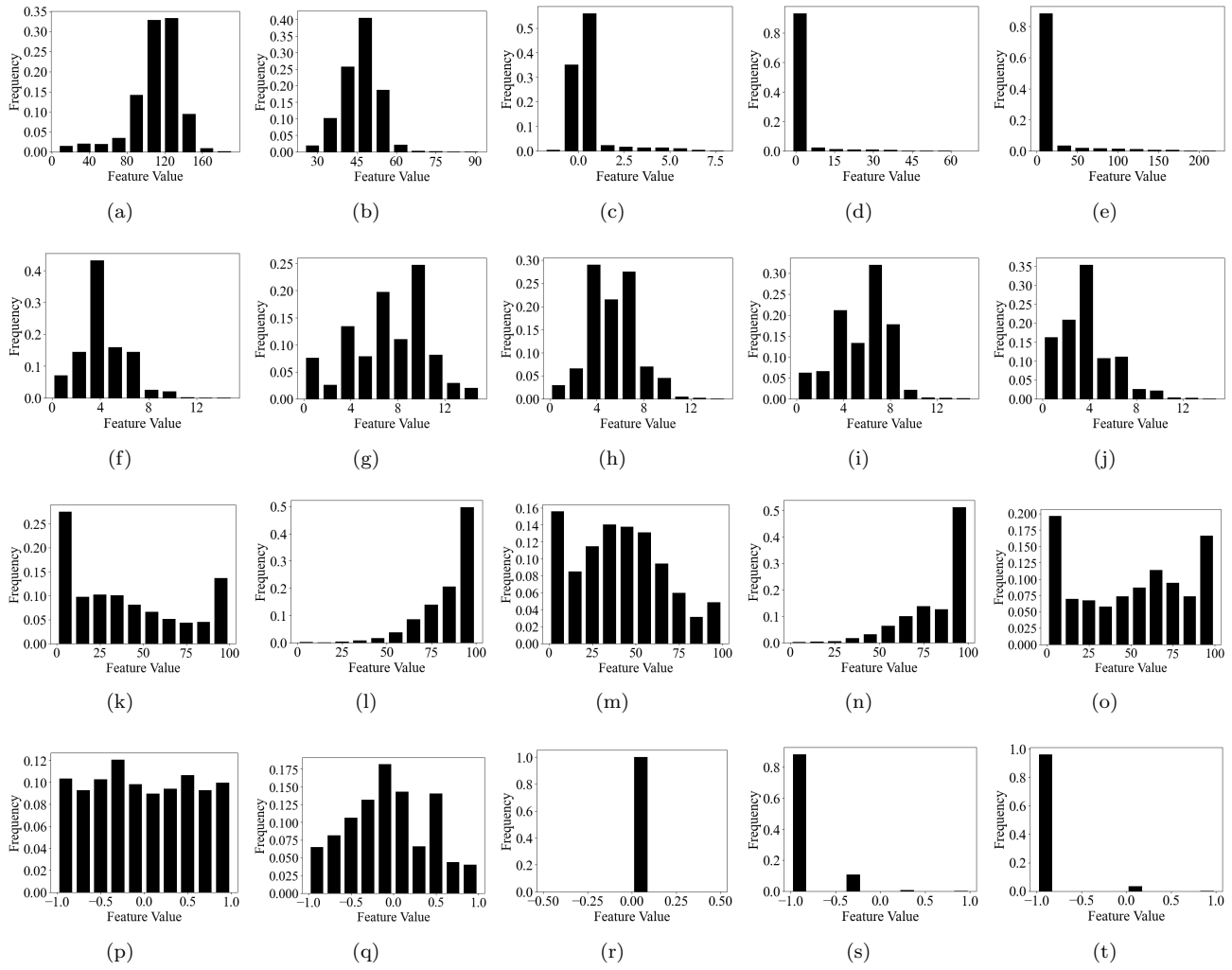


Fig.2. Visualization of histogram distributions. We plot the first five features on five complete datasets and we plot the histogram distribution of each feature. Blue represents the first feature dimension, red represents the second feature dimension, orange represents the third feature dimension, and so on. (a)–(e) Feature 1–5 histogram distribution on HTRU2. (f)–(j) Feature 1–5 histogram distribution on letter. (k)–(o) Feature 1–5 histogram distribution on pendigits. (p)–(t) Feature 1–5 histogram distribution on segment.

4.1.2 Implementation Details

The proposed RSS algorithm is compared with the following methods: 1) Mean: missing features are replaced with mean values of those features computed for all training samples; 2) KNN: missing features are replaced with mean values of those features from the k nearest training samples (we set $k = 5$); 3) MICE[2]: iterative filling of missing attributes using Multiple-Imputation by Chained Equation (MICE), where several imputations are drawn from the conditional distribution of data by Markov chain Monte Carlo techniques. We train five different models on five imputed datasets generated by MICE and combine their results. 4) GMM[7]: missing features are replaced with values sampled from Gaussian Mixture Models (GMM) estimated from incomplete data using the EM algorithm. 5) Softimpute[23]: matrix completion by iterative soft thresholding of SVD decompositions; 6) AFASMC[24]: active feature acquisition with supervised matrix completion. Since the last two matrix completion methods treat the training matrix as a whole to complete based on low-rank assumptions and cannot be applied when a test instance

contains missing values, we don't include them in Table 2 and Table 4.

We use NRS as our classifier for all methods above in our experiments. Following the settings in [21], we build NRS by one depthwise convolution layer and two FC layers with batch normalization (BN)[25]. We set $nPer = 1$, $dH/dW = 3$ and only set different $nMul$ for these datasets following the suggestions in [21], as shown in Table 1. For RSS, we have two extra hyperparameters and we set the number of bins $M = 10$ and the test ensemble size $H = 20$ in all experiments unless otherwise specified. We split 10% of the training data for validation to determine the total epochs separately for each dataset. All methods are trained under the same setting: NRS are trained for 20–50 epochs, using Adam[26] as the optimizer and initializing the learning rate to 1e-4.

To further validate the choice of the NRS architecture and confirm the effectiveness of our method, we also compare RSS with other classification algorithms, e.g., logistic regression, MLP and random forests[18] using KNN, Mean and MICE imputation in Subsection 4.5. All our experiments were conducted using PyTorch on Tesla M40 GPUs and we will make our code publicly available.

4.2 Results on Classification Performance

Table 2 shows the average and standard deviation of classification accuracy on the first six incomplete datasets containing natural missing values. Table 3 and Table 4 show the full test accuracy and missing test accuracy on the last seven complete datasets with four levels of missing values, respectively. To compare the classification performance, paired t-tests at 95% confidence interval are used to compare the classification achieved by RSS with the other methods in both tables.

Table 2 shows that RSS achieves significantly better classification accuracy (under paired t-tests) than the other methods on kidney, horse, pima and bands, where the first three have relatively high levels of missing values. However, there is no significant difference between RSS and the other methods on mammographics and hepatitis, where the level of missing values is low.

Table 3 and Table 4 show that RSS achieves the highest accuracy for most of the times in both missing test and full test situations with the datasets containing artificial missing values. As can be seen, our RSS method significantly outperforms other methods, since the win/tie/lose counts show that our RSS wins for most times and seldom loses. It demonstrates the effectiveness of RSS across datasets with various dimensionalities and sizes, and under different levels of missing values. Also, RSS has a larger edge over other methods along with the increase of the portion of missing values, which indicates that our method is effective to cope with datasets containing numerous missing values.

It is clear from the results that RSS is generally better than simple-imputation methods (Mean and KNN), matrix completion methods (Softimpute and AFASMC) and model-based method (GMM), which shows that our method reflects better uncertainty and gets more reliable estimations than these methods. Furthermore, RSS also gets better performance than multiple-imputation method (MICE), which indicates that the inherent random subspace method together with the dynamic training and ensemble testing strategy is effective. We will further study the effectiveness of each component in RSS through ablation studies and the sensitivity of hyperparameters in hyperparameter studies in the next two subsections. The accuracy curves of different methods on different datasets during training are shown in Fig. 3. We can see that our RSS converges well and the injected randomness during training does not affect the convergence stability of RSS.

Table 2. Accuracy (%) on the First Six Incomplete Datasets

Methods	Mammo.	Hepatits	Kidney	Horse	Pima	Bands	Win/Tie/Lose
RSS	82.2±1.7	**82.3±5.1**	**97.3±1.4**	**82.3±3.5**	**75.7±2.3**	**70.4±2.7**	
Mean	81.6±1.9	82.1±5.6	96.9±1.5	80.3±3.2•	74.3±2.8•	68.4±3.2•	3/3/0
KNN[10]	81.9±1.4	82.0±4.9	95.6±1.7•	80.6±3.5•	75.6±2.6	68.3±3.0•	3/3/0
MICE[2]	**82.4±1.7**	81.9±5.1	95.9±1.4•	80.8±3.6•	75.6±2.6	68.5±3.1•	3/3/0
GMM[7]	82.1±1.7	81.6±6.0	96.1±1.8•	78.9±3.5•	74.1±2.5•	68.0±3.3•	4/2/0

Note: We report the average accuracy and standard deviation of 100 trials. •/○ indicates that our RSS is significantly better/worse than the corresponding method (pairwise t-tests at 95% significance level).

Table 3. Full Test Accuracy (%) on the Last Seven Complete Datasets

Methods	PMV	Chess-krkp	Chess-krkopt	Letter	HTRU2	Yeast	Segment	Pendigts	Win/Tie/Lose
RSS	20%	98.8±0.4	**67.2±0.6**	**95.5±0.3**	**97.9±0.2**	59.6±1.9	95.8±0.8	**99.3±0.1**	
Mean		98.5±0.6	60.4±0.6•	94.3±0.4•	97.8±0.2	58.0±1.8•	95.1±1.3•	99.1±0.2•	5/2/0
KNN[10]		98.6±0.5•	57.7±0.8•	95.4±0.3	**97.9±0.1**	58.6±1.6	**96.6±0.8**○	99.2±0.1•	3/3/1
MICE[2]		**99.2±0.4**○	63.6±0.7•	94.6±0.4•	97.8±0.2	59.4±1.4	96.1±0.7	**99.3±0.1**	2/4/1
Softimpute[23]		**99.2±0.4**○	62.5±0.7•	94.8±0.4•	**97.9±0.2**	**59.7±1.7**	95.8±0.6	**99.3±0.1**	2/4/1
AFASMC[24]		99.1±0.2○	62.8±0.8•	93.9±1.2•	**97.9±0.1**	58.2±2.0•	96.4±1.0○	98.9±0.1•	4/1/2
RSS	40%	97.4±0.6	**56.3±0.6**	**91.7±0.6**	**97.5±0.2**	**57.5±3.0**	93.9±1.1	**98.9±0.2**	
Mean		96.8±0.6•	46.0±1.0•	88.6±0.4•	**97.5±0.2**	56.2±2.6•	91.9±1.1•	98.6±0.2•	6/1/0
KNN[10]		95.7±0.7•	41.1±0.8•	87.2±0.6•	**97.5±0.2**	56.4±3.0	94.5±1.0	98.4±0.2•	4/3/0
MICE[2]		97.4±0.6	48.2±1.1•	88.2±0.6•	97.4±0.3•	56.9±3.3	93.8±0.8	98.8±0.3	3/4/0
Softimpute[23]		97.8±0.4	48.5±0.7•	89.9±0.4•	**97.5±0.3**	57.2±3.1	94.2±1.3	98.6±0.2•	3/4/0
AFASMC[24]		**98.1±0.3**○	48.8±1.0•	85.8±0.7•	97.4±0.3•	56.7±3.0	**94.8±1.4**	97.5±0.5•	4/2/1
RSS	60%	**96.0±0.8**	**46.9±0.9**	**84.7±0.8**	**97.5±0.1**	**55.2±2.3**	**92.0±1.7**	**97.7±0.4**	
Mean		93.6±1.0•	35.4±0.8•	76.7±0.9•	97.3±0.2•	53.9±1.9	87.6±3.6•	97.3±0.4•	6/1/0
KNN[10]		91.0±1.2•	32.1±1.0•	69.5±0.9•	97.3±0.3•	51.7±2.5•	85.8±2.8•	94.5±1.2•	7/0/0
MICE[2]		93.6±1.6•	36.1±1.6•	78.4±1.1•	97.3±0.3•	52.4±2.7•	89.7±2.1•	96.9±0.8•	7/0/0
Softimpute[23]		94.6±0.5•	37.3±1.0•	78.7±1.2•	97.3±0.2•	48.1±2.7•	91.1±1.4	96.8±0.6•	6/1/0
AFASMC[24]		93.5±1.3•	37.2±0.9•	66.4±2.6•	97.4±0.2	54.2±2.0	91.6±1.4	93.3±1.0•	4/3/0
RSS	70%	**94.2±1.0**	**42.1±1.1**	**78.6±0.4**	**97.3±0.3**	**55.5±1.4**	**89.3±3.1**	**96.1±0.9**	
Mean		90.9±2.1•	30.1±1.2•	68.5±1.1•	97.1±0.5•	51.9±2.8•	85.5±2.9•	95.9±0.7	6/1/0
KNN[10]		86.6±1.9•	26.1±0.8•	56.6±1.7•	97.2±0.3•	49.6±2.9•	81.4±2.3•	90.7±1.2•	7/0/0
MICE[2]		89.0±1.6•	31.3±1.5•	66.7±1.8•	96.9±0.3•	51.0±4.1•	84.6±3.1•	94.3±0.9•	7/0/0
Softimpute[23]		92.7±1.2•	31.4±1.3•	66.0±1.6•	97.1±0.3•	45.0±1.8•	83.0±2.9•	94.5±0.7•	7/0/0
AFASMC[24]		89.6±1.8•	30.9±1.9•	50.8±3.4•	97.2±0.2•	51.8±1.8•	87.5±1.7•	86.9±2.2•	7/0/0

Note: We report the average accuracy and standard deviation of 10 trials. •/○ indicates that our RSS is significantly better/worse than the corresponding method (pairwise t-tests at 95% significance level).

We also carefully study the running speed of RSS, simple-imputation methods (Mean and KNN) and multiple-imputation method (MICE) in Subsection 4.6. The results show that RSS greatly saves the training costs when compared to MICE, especially when the feature dimensionality is high.

4.3 Ablation Studies

In this subsection, we conduct ablation studies on the three components in RSS on three datasets, i.e., chess-krkp, HTRU-2 and letter:

1. Dynamic training (DR): RSS resamples missing values dynamically at each epoch. For comparisons, we also study the static strategy where we only sample at the first epoch and then fix them during training.

2. Ensemble testing (ET): RSS runs the model multiple times and ensemble the results to get the final prediction and we also ablate this strategy.

3. NRS architecture: The above two strategies are not limited to NRS and can also be applied to other architectures, e.g., MLP. We also conduct experiments for MLP to further investigate our algorithm. In this experiment, we adopt a typical

Table 4. Missing Test Accuracy (%) on the Last Seven Complete Datasets

Methods	PMV	Chess-krkp	Chess-krkopt	Letter	HTRU2	Yeast	Segment	Pendigts	Win/Tie/Lose
RSS	20%	95.9±0.7	**47.7±0.7**	84.6±0.5	**97.7±0.1**	52.7±1.9	93.5±1.0	96.4±0.2	
Mean		94.8±1.0•	43.8±0.7•	81.0±0.5•	97.5±0.2•	53.0±2.6	89.3±0.8•	95.6±0.2•	6/1/0
KNN[10]		95.3±0.6•	40.3±0.5•	**87.4±0.5**○	97.4±0.2•	51.1±1.7•	**94.1±0.8**	**97.9±0.3**○	4/1/2
MICE[2]		**97.2±0.4**○	47.1±0.5	83.1±0.7•	97.6±0.2	**54.7±1.9**	93.8±0.6	97.2±0.3○	1/4/2
RSS	40%	88.3±1.0	**32.4±0.6**	**68.8±0.5**	**97.1±0.2**	**46.9±2.6**	**87.2±1.4**	89.7±0.5	
Mean		86.8±0.8•	28.3±0.6•	62.3±0.4•	**97.1±0.2**	45.7±2.6	81.9±1.7•	87.3±0.6•	5/2/0
KNN[10]		86.0±1.2•	24.2±0.5•	51.1±1.0•	96.6±0.2•	43.3±2.2•	79.8±1.5•	81.6±0.7•	7/0/0
MICE[2]		**90.9±0.9**○	28.7±0.6•	61.8±0.9•	**97.1±0.2**	46.5±2.1	86.2±1.7	**90.7±0.3**○	2/3/2
RSS	60%	**80.6±0.9**	**23.7±0.6**	**48.1±0.4**	**96.6±0.2**	**40.8±2.7**	**76.0±1.4**	**76.1±0.9**	
Mean		77.5±1.3•	19.9±0.6•	41.4±0.6•	96.0±0.3•	40.3±1.9	71.1±1.2•	73.1±0.9•	6/1/0
KNN[10]		72.5±1.0•	17.9±0.6•	26.3±0.7•	95.3±0.3•	36.6±1.9•	53.6±2.3•	59.7±0.9•	7/0/0
MICE[2]		80.5±0.8	19.7±0.6•	40.7±0.6•	96.2±0.2•	40.2±2.2	75.0±2.6	74.7±1.0•	4/3/0
RSS	70%	**75.3±1.2**	**21.0±0.6**	**35.8±0.7**	**95.5±0.3**	**39.2±1.6**	**66.2±1.4**	**65.0±0.8**	
Mean		71.5±1.1•	17.3±0.5•	30.7±0.8•	95.2±0.3•	37.7±2.0	63.6±1.9•	62.6±0.9•	6/1/0
KNN[10]		66.1±1.8•	15.7±0.5•	18.6±0.4•	94.7±0.3•	34.4±1.8•	44.6±1.7•	49.7±0.9•	7/0/0
MICE[2]		72.7±1.3•	17.6±0.3•	29.6±0.6•	95.1±0.2•	37.9±1.9	64.7±3.1	61.7±1.2•	5/2/0

Note: We report the average accuracy and standard deviation of 10 trials. •/○ indicates that our RSS is significantly better/worse than the corresponding method (pairwise t-tests at 95% significance level).

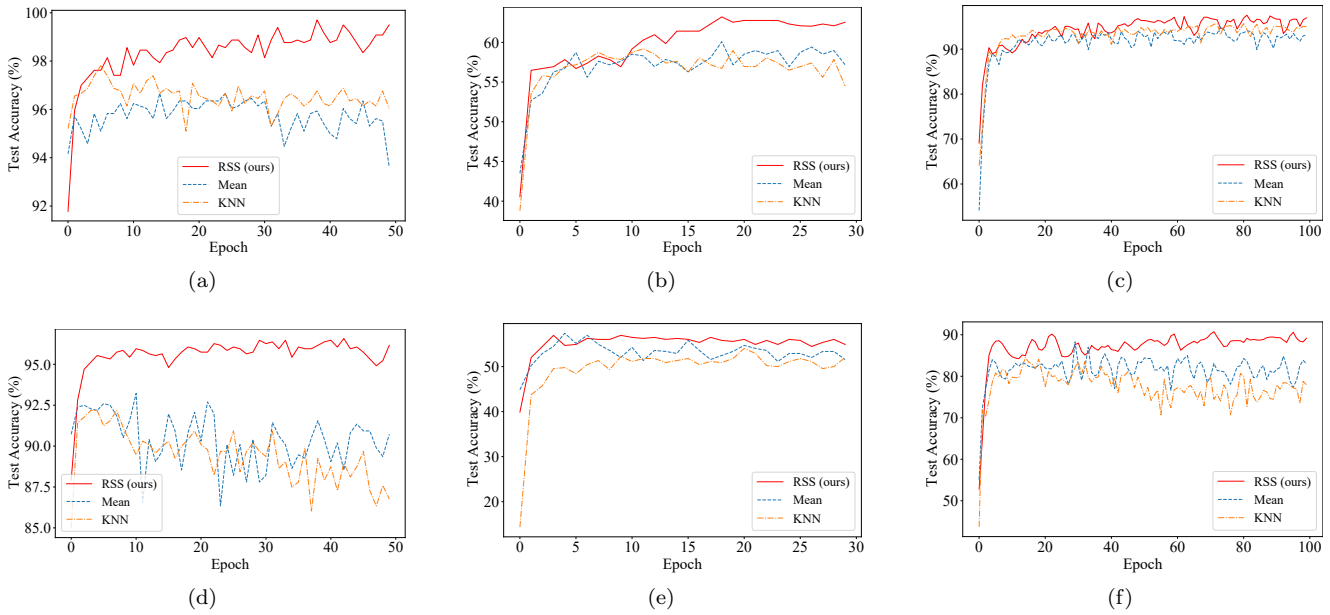


Fig.3. Full test accuracy curve on chess-krkp, yeast and segment under different PMV. (a) PMV=40% on chess-krkp. (b) PMV=40% on yeast. (c) PMV=40% on segment. (d) PMV=70% on chess-krkp. (e) PMV=70% on yeast. (f) PMV=70% on segment.

MLP with two ReLU hidden layers and we also use batch-normalization for fair comparisons.

The experimental results are shown in Table 5 and we can have the following observations:

1) Both strategies DR and ET are effective in RSS. From the vanilla baseline case 1, either by using DR (case 2) or ET (case 3), we can get higher accuracy on all the three datasets consistently. Finally, by combining both strategies (case 4), we achieve the best performance on all these datasets.

2) The above two strategies are generalizable. Notice that when we use MLP as our classifier, we are operating only on the original space and we can compare the performance of histogram random sampling to fill in the vacant values instead of multiple subspaces. Case 6 directly uses histogram random sampling to

Table 5. Ablation Studies Using Different Strategies at Different Levels of Missing Values

Case	Model	Scheme		Chess-krkp				HTRU2				Letter			
		DR	ET	20%	40%	60%	70%	20%	40%	60%	70%	20%	40%	60%	70%
1	NRS	×	×	92.9±0.6	83.4±1.0	73.7±1.2	68.2±1.3	96.6±0.2	95.5±0.3	94.0±0.3	93.2±0.3	73.1±1.2	48.9±1.0	27.7±0.7	18.6±0.3
2		✓	×	94.3±0.5	87.0±1.0	78.4±1.0	73.4±1.6	97.3±0.1	96.9±0.2	95.7±0.2	94.8±0.2	82.9±0.6	66.1±0.5	44.9±0.4	33.1±0.6
3		×	✓	94.4±0.5	87.4±0.7	78.9±1.0	74.1±1.6	97.0±0.2	96.3±0.2	95.0±0.3	94.0±0.3	78.3±0.9	58.3±2.3	37.9±1.3	27.4±0.6
4		✓	✓	**95.9±0.7**	**88.3±1.0**	**80.6±0.9**	**75.3±1.2**	**97.7±0.1**	**97.1±0.2**	**96.6±0.2**	**95.5±0.3**	**84.6±0.5**	**68.8±0.5**	**48.1±0.4**	**35.8±0.7**
5	MLP	×	×	88.3±1.0	77.6±1.7	66.3±1.1	59.8±1.7	96.9±0.2	95.9±0.2	93.7±0.2	92.5±0.3	63.5±0.9	37.4±0.7	18.1±0.5	11.7±0.4
6		✓	×	89.7±0.8	80.9±1.3	70.5±1.5	65.9±1.0	96.9±0.1	95.9±0.2	93.8±0.3	92.5±0.3	68.8±0.5	45.0±0.5	24.9±0.4	17.3±0.5
7		×	✓	92.2±0.5	84.7±0.9	74.7±1.3	68.5±1.8	**97.2±0.1**	**96.5±0.3**	94.4±0.3	**93.2±0.3**	77.0±0.6	55.3±0.5	31.8±0.6	21.3±0.6
8		✓	✓	**93.7±0.3**	**87.3±0.5**	**78.0±1.0**	**73.5±1.6**	**97.2±0.1**	96.4±0.2	**94.5±0.3**	93.1±0.4	**80.1±0.5**	**60.4±0.6**	**37.8±0.6**	**27.6±0.7**

Note: We report the missing test accuracy (%) here.

fill in the missing values in the original feature vector while case 5 serves as the baseline. As can be seen, histogram random sampling (our DR strategy) also works when filling in the original feature space. We also achieve the highest accuracies for MLP when combining both strategies on chess-krkp and letter. It indicates that our strategies can also be applied to other architectures.

3) NRS serves as a strong baseline classifier. NRS achieves better performance than MLP consistently under all settings on all datasets. As introduced before, we use NRS as our classifier for all comparison methods and we now show that NRS serves as a strong baseline classifier.

4) The inherent random subspace method is crucial in our algorithm. By comparing case 4 with case 1 and case 8 with case 5, the NRS architecture has more improvements than MLP when combined with the two strategies, especially on HTRU2. It indicates that the inherent random subspace method allows us to use multiple values to estimate one missing feature in different random subspaces, which provides us more robust and accurate estimations.

In short, the DR and ET strategies are effective for both NRS and MLP. Combining these two strategies with NRS has more improvements than MLP and finally achieves the best performance on all these three datasets, which explains again why we adopt NRS in this paper.

4.4 Hyperparameters Studies

In this subsection, we study the sensitivity of hyperparameters in RSS, namely, the test ensemble size H, the depth expansion rate $nMul$ and the number of bins M in the histogram. The experimental results are shown in Table 6 and for better illustration we plot the corresponding figures on HTRU2 in Fig. 4.

Ensemble Size H. Here we vary H and keep other settings the same as before. Table 6 and Fig. 4(a) show that when H grows, the accuracy also gets higher, which indicates that we can get more accurate predictions by enlarging the ensemble size in our ET strategy. It is also worth mentioning that in contrast to other ensemble strategies, e.g., MICE, the model size in our RSS remains unchanged as we increase H, which saves a lot of computing and storage overhead.

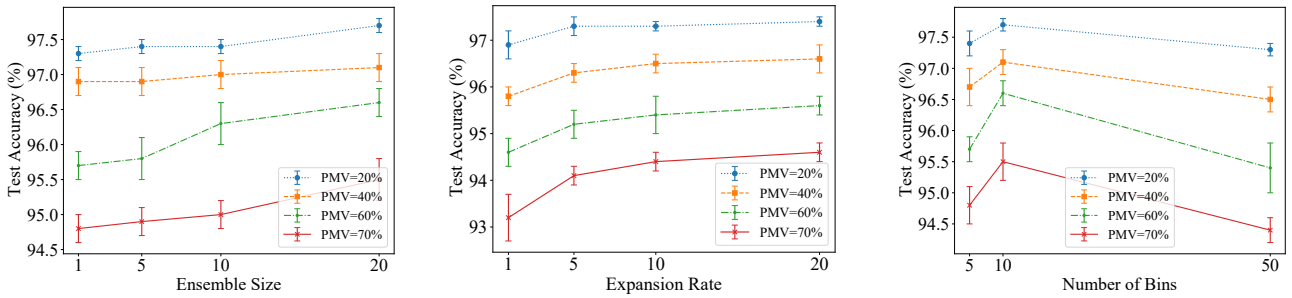
Expansion Rate $nMul$. Here we vary $nMul$ and other settings remain unchanged. The results in Table 6 and Fig. 4(b) show that when $nMul$ grows, the average accuracy increases and the standard deviation becomes smaller. It indicates that as $nMul$ grows, more random subspaces and base learners are integrated into our model, hence the estimation gets more robust and accurate and the performance becomes better.

Number of Bins M. As known in density estimation methods, the value of M in histogram distribution plays an important role and we also study the sensitivity of M here. We only vary M and other settings remain un-

Table 6. Hyperparameters Studies at Different Levels of Missing Values

		Chess-krkp				HTRU2				Letter			
		20%	40%	60%	70%	20%	40%	60%	70%	20%	40%	60%	70%
H	1	95.3±0.5	87.0±1.0	78.4±1.0	73.4±1.6	97.3±0.1	96.9±0.2	95.7±0.2	94.8±0.2	82.9±0.6	66.1±0.5	44.9±0.4	33.1±0.6
	5	95.8±0.7	87.9±1.0	80.2±0.9	75.1±1.4	97.4±0.1	96.9±0.2	95.8±0.3	94.9±0.2	84.3±0.6	68.5±0.4	47.2±0.4	35.4±0.7
	10	**95.9±0.7**	88.1±0.9	**80.6±0.8**	**75.4±1.2**	97.4±0.1	97.0±0.2	96.3±0.3	95.0±0.2	84.5±0.5	68.6±0.5	47.8±0.3	35.7±0.7
	20	**95.9±0.7**	**88.3±1.0**	**80.6±0.9**	75.3±1.2	**97.7±0.1**	**97.1±0.2**	**96.6±0.2**	**95.5±0.3**	**84.6±0.5**	**68.8±0.5**	**48.1±0.4**	**35.8±0.7**
$nMul$	1	93.4±1.0	86.2±1.4	78.3±1.7	73.4±1.2	96.9±0.3	95.8±0.2	94.6±0.3	93.2±0.5	76.9±1.0	56.0±1.0	35.0±0.8	24.9±0.6
	5	94.8±1.0	**88.7±1.0**	81.3±1.0	76.0±1.2	97.3±0.2	96.3±0.2	95.2±0.3	94.1±0.2	81.9±0.7	63.9±0.3	41.8±0.8	30.5±0.4
	10	95.7±0.8	88.6±1.1	**81.4±1.1**	**76.3±1.4**	97.3±0.1	96.5±0.2	95.4±0.4	94.4±0.2	83.4±0.7	66.1±0.5	44.0±0.8	32.4±0.6
	20	**95.9±0.7**	88.3±1.0	80.6±0.9	75.3±1.2	**97.4±0.1**	**96.6±0.3**	**95.6±0.2**	**94.6±0.2**	**83.9±0.5**	**67.7±0.7**	**46.3±0.9**	**34.1±0.7**
M	5	95.6±0.7	88.0±1.1	79.7±0.7	74.8±1.2	97.4±0.2	96.7±0.3	95.7±0.2	94.8±0.3	**84.7±0.6**	68.7±0.3	47.9±0.8	35.9±0.7
	10	**95.9±0.7**	88.3±1.0	80.6±0.9	75.3±1.2	**97.7±0.1**	**97.1±0.2**	**96.6±0.2**	**95.5±0.3**	84.6±0.5	**68.8±0.5**	**48.1±0.4**	35.8±0.7
	50	**95.9±1.0**	**88.4±1.0**	**80.9±1.1**	**76.0±1.2**	97.3±0.1	96.5±0.2	95.4±0.4	94.4±0.2	84.5±0.6	**68.8±0.6**	47.8±0.7	**36.1±0.6**

Note: We report the missing test accuracy (%) here.



(a) (b) (c)

Fig.4. Hyperparameters studies of H, $nMul$ and M on HTRU2. We plot the average accuracy and standard deviation of 10 trials at each point. (a) Ensemble size H. (b) Expansion rate $nMul$. (c) Number of bins M.

changed. As can be seen, the optimal value of M varies on different datasets, e.g., M equals 10 achieves the highest accuracy on HTRU2 while M equals 50 performs best on chess-krkp. As mentioned before, we directly set M to 10 for all datasets in this paper and it indicates that we can get better performance by choosing more appropriate M on each dataset.

4.5 Comparisons among Different Classifiers

In the previous subsections, we used NRS as the classifier for our RSS as well as other comparative imputation methods and we trained them under the same setting. To further validate the effectiveness of our RSS, we also compare RSS with other classification algorithms: logistic regression (LR), MLP and random forests (RF) using Mean, KNN and MICE imputation. As in Subsection 4.2, we report both the full test accuracy and missing test accuracy in Table 7 and Table 8, respectively. From Table 7 and Table 8 we can have the following conclusions:

1) There is no best classification algorithm in all cases. For example, MLP achieves higher accuracies than RF on chess-krkp when using the same imputation method but performs worse on letter.

2) There is no best imputation algorithm in all cases. For example, KNN imputation performs the best among the three imputation methods on letter, segment and pendigits under low missing levels ($PMV = 20\%$) but achieves the lowest accuracy when the missing level increases. Also, we can see that when using the same classification algorithm, Mean imputation sometimes performs better than MICE and sometimes performs worse, depending on the specific dataset and the level of missing rate.

3) Overall, our RSS method significantly outperforms

Table 7. Full Test Accuracy (%) of Different Classifiers

Methods	PMV	Chess-krkp	Chess-krkopt	Letter	HTRU2	Yeast	Segment	Pendigits	Win/Tie/Lose
RSS		98.8±0.4	**67.2±0.6**	**95.5±0.3**	97.9±0.2	59.6±1.9	95.8±0.8	**99.3±0.1**	
LR+Mean		95.4±0.7•	31.8±0.5•	70.7±0.4•	97.7±0.1	54.8±1.2•	89.2±1.2•	88.4±0.5•	6/1/0
LR+KNN[10]		95.5±0.8•	31.8±0.4•	75.5±0.4•	97.7±0.2	53.8±1.2•	91.6±1.0•	93.5±0.3•	6/1/0
LR+MICE[2]		95.9±0.6•	31.7±0.4•	73.6±0.4•	97.9±0.2	54.6±1.0•	90.3±1.1•	92.9±0.4•	6/1/0
MLP+Mean	20%	98.6±0.5	58.3±0.8•	93.2±0.5•	97.8±0.2	58.6±1.8	93.8±1.1•	99.1±0.2•	4/3/0
MLP+KNN[10]		98.6±0.4	56.8±0.8•	94.8±0.3•	97.8±0.2	58.3±1.5	95.8±0.7	99.3±0.1	2/5/0
MLP+MICE[2]		**99.1±0.3**○	60.6±0.9•	94.0±0.5•	97.8±0.1	59.4±1.9	94.8±0.9•	99.2±0.2	3/3/1
RF+Mean		97.6±0.5•	57.6±0.6•	93.4±0.3•	97.9±0.1	60.0±2.6	96.5±0.9○	98.6±0.2•	4/2/1
RF+KNN[10]		98.5±0.4•	56.1±0.6•	94.6±0.4•	97.9±0.2	59.0±1.8	**97.2±0.8**○	98.9±0.1•	4/2/1
RF+MICE[2]		98.0±0.7•	60.5±0.8•	93.6±0.4•	**98.0±0.1**	**60.4±2.0**	96.5±0.7○	98.8±0.2•	4/2/1
RSS		97.4±0.6	**56.3±0.6**	**91.7±0.6**	97.5±0.2	**57.5±3.0**	93.9±1.1	**98.9±0.2**	
LR+Mean		94.4±0.6•	30.8±0.4•	64.2±0.5•	97.3±0.2	53.0±2.7•	85.4±1.0•	84.7±0.7•	6/1/0
LR+KNN[10]		94.4±0.7•	30.5±0.6•	66.9±0.6•	97.3±0.2	51.0±2.8•	89.9±1.4•	86.8±0.5•	6/1/0
LR+MICE[2]		94.9±0.6•	31.1±0.4•	67.5±1.4•	97.5±0.2	50.9±2.7•	89.4±1.2•	90.0±0.6•	6/1/0
MLP+Mean	40%	97.1±0.5	47.6±0.8•	86.7±0.6•	97.4±0.2	57.3±2.6	90.9±2.3•	98.5±0.3•	4/3/0
MLP+KNN[10]		96.1±1.0•	43.5±0.6•	86.6±0.7•	97.5±0.2	56.4±3.1	93.8±1.4	98.6±0.2•	4/3/0
MLP+MICE[2]		**97.9±0.7**	48.3±1.0•	87.5±0.7•	97.5±0.3	56.6±3.1	93.3±1.0	98.7±0.2	2/5/0
RF+Mean		95.7±1.1•	48.3±0.7•	89.1±0.5•	**97.7±0.2**	57.3±1.7	95.5±1.0○	97.8±0.2•	4/2/1
RF+KNN[10]		97.0±0.8	46.3±0.8•	88.3±0.6•	**97.7±0.2**	54.9±2.0•	**96.0±0.8**○	98.0±0.2•	4/2/1
RF+MICE[2]		96.5±0.7•	50.2±0.7•	88.6±0.6•	**97.7±0.2**	57.1±2.1	95.1±1.0○	98.2±0.2•	4/2/1
RSS		**96.0±0.8**	**46.9±0.9**	**84.7±0.8**	97.5±0.1	**55.2±2.3**	92.0±1.7	**97.7±0.4**	
LR+Mean		93.6±1.0•	30.0±0.6•	58.4±0.5•	96.3±0.2•	47.9±1.6•	79.9±1.6•	83.0±0.5•	7/0/0
LR+KNN[10]		91.7±1.1•	28.7±0.8•	57.9±1.4•	97.3±0.2•	45.3±3.1•	81.8±1.9•	83.1±0.5•	7/0/0
LR+MICE[2]		93.9±1.2•	29.9±0.5•	61.5±1.7•	97.5±0.4	46.1±1.1•	87.4±1.9•	84.9±1.4•	6/1/0
MLP+Mean	60%	95.2±0.8•	38.6±0.9•	75.9±1.5•	97.4±0.2	54.3±2.3	85.5±3.8•	96.8±0.6•	5/2/0
MLP+KNN[10]		91.7±0.9•	33.8±1.0•	71.5±0.9•	97.4±0.2	52.6±2.1•	86.1±2.2•	96.3±0.4•	6/1/0
MLP+MICE[2]		94.9±0.8•	38.2±1.0•	77.6±1.0•	97.3±0.3	52.7±1.4•	89.7±1.8•	96.9±0.5•	6/1/0
RF+Mean		95.0±1.0•	40.6±0.8•	81.6±0.6•	97.8±0.1○	53.9±1.6	**93.9±1.0**○	95.5±0.5•	4/1/2
RF+KNN[10]		95.0±1.1•	37.6±0.9•	77.5±1.1•	97.6±0.1	50.0±3.0•	92.5±0.7	94.2±0.8•	5/2/0
RF+MICE[2]		95.0±1.0•	40.5±0.9•	80.5±0.8•	**97.8±0.1**○	52.5±2.4•	92.8±1.3	96.4±0.4•	5/1/1
RSS		94.2±1.0	**42.1±1.1**	**78.6±0.4**	97.3±0.3	**55.5±1.4**	89.3±3.1	**96.1±0.9**	
LR+Mean		92.8±1.1•	29.8±0.5•	55.1±0.7•	95.2±0.6•	48.6±1.7•	78.2±1.2•	81.9±0.6•	7/0/0
LR+KNN[10]		89.9±1.5•	27.6±0.7•	53.4±1.1•	97.1±0.2	45.9±2.7•	80.3±1.6•	81.5±0.9•	6/1/0
LR+MICE[2]		92.7±1.2•	29.4±0.7•	55.1±2.1•	96.5±1.9•	45.7±1.3•	83.3±2.4•	82.2±1.0•	7/0/0
MLP+Mean	70%	93.0±1.2•	33.1±0.8•	68.2±0.8•	97.2±0.4	53.7±1.5•	78.5±4.2•	95.8±0.7	5/2/0
MLP+KNN[10]		88.5±2.6•	28.2±0.8•	59.4±1.0•	97.3±0.2	52.0±1.5•	83.2±1.4•	93.2±0.8•	6/1/0
MLP+MICE[2]		91.3±1.5•	33.7±1.5•	66.5±2.5•	97.1±0.3	52.1±3.5•	85.0±2.3•	94.5±0.9•	6/1/0
RF+Mean		**94.6±0.7**	36.6±0.6•	75.8±0.9•	**97.7±0.2**○	52.9±2.4•	**92.5±1.0**○	93.4±0.8•	4/1/2
RF+KNN[10]		93.3±1.5•	33.2±0.6•	70.4±1.0•	97.3±0.3	50.4±2.0•	90.0±1.5	91.5±0.7•	5/2/0
RF+MICE[2]		94.5±0.6	36.1±0.8•	73.7±1.2•	97.5±0.2	49.8±3.3•	90.8±2.2	93.0±0.7•	4/3/0

Note: We report the average accuracy and standard deviation of 10 trials. •/○ indicates that our RSS is significantly better/worse than the corresponding method (pairwise t-tests at 95% significance level).

other methods, since the win/tie/lose counts show that our RSS wins for most times and seldom loses and it further demonstrates the effectiveness of our method in these comparisons.

4.6 Speed Comparisons

In this subsection, we compare the running speed of our RSS with other imputation methods: Mean, KNN and MICE. Notice that the feature dimensionality has a great influence on the running speed of different methods. Hence, we evaluate different methods on one low-dimensional dataset yeast (8-dimensional) as well as two more datasets with higher dimensionalities: dna (180-dimensional) and protein (357-dimensional). We also take the influence of the proportion of missing values (PMV) into consideration and we set PMV to 20% and 60% for each dataset. All the other training settings remain the same as before. For the training/inference time, we count the total running time, which equals the sum of preprocessing time (imputation time) and the network running time (network total training/inference time). We report the average training/inference time and average accuracy of 10 trials in Table 9 (For accuracy, we also report the standard deviation).

Table 8. Missing Test Accuracy (%) of Different Classifiers

Methods	PMV	Chess-krkp	Chess-krkopt	Letter	HTRU2	Yeast	Segment	Pendigits	Win/Tie/Lose
RSS		95.9±0.7	**47.7±0.7**	84.6±0.5	97.7±0.1	52.7±1.9	93.5±1.0	96.4±0.2	
LR+Mean		90.8±0.6●	28.1±0.5●	55.8±0.6●	97.3±0.2●	50.1±1.7●	82.6±1.7●	81.2±0.3●	7/0/0
LR+KNN[10]		92.6±0.7●	27.5±0.4●	70.5±0.8●	97.2±0.2●	48.8±1.9●	89.9±1.1●	91.9±0.6●	7/0/0
LR+MICE[2]		94.1±0.7●	28.8±0.4●	62.1±0.7●	97.6±0.1	50.2±2.2●	87.3±1.0●	87.4±0.4●	6/1/0
MLP+Mean	20%	94.1±0.7●	43.2±0.6●	78.9±0.6●	97.4±0.1●	52.6±2.0	87.5±0.7●	95.0±0.4●	6/1/0
MLP+KNN[10]		95.2±0.5●	40.4±0.7●	86.5±0.6○	97.4±0.1●	51.6±1.8	93.7±1.0	**97.7±0.3**○	3/2/2
MLP+MICE[2]		**97.2±0.5**○	46.2±0.6●	82.4±0.8●	97.7±0.2	53.9±1.8	92.1±0.7●	97.1±0.3○	3/2/2
RF+Mean		95.0±0.5●	43.7±0.5●	84.4±0.4	97.7±0.2	53.1±1.5	93.5±0.6	95.9±0.4●	3/4/0
RF+KNN[10]		95.6±0.5	41.2±0.6●	**87.8±0.6**○	97.6±0.2	51.2±0.9●	**95.1±1.0**○	97.5±0.3○	2/2/3
RF+MICE[2]		96.5±0.8○	47.0±0.7●	83.9±0.6●	**97.8±0.1**	**54.2±2.6**	94.3±0.6○	96.8±0.2○	2/2/3
RSS		88.3±1.0	**32.4±0.6**	**68.8±0.5**	97.1±0.2	46.9±2.6	87.2±1.4	89.7±0.5	
LR+Mean		85.7±0.7●	24.9±0.5●	42.4±0.5●	96.8±0.2●	43.6±2.5●	74.8±1.2●	71.4±0.5●	7/0/0
LR+KNN[10]		85.8±0.8●	23.2±0.5●	41.0±0.6●	96.6±0.2●	41.0±1.8●	76.6±1.5●	70.4±0.7●	7/0/0
LR+MICE[2]		88.7±1.0	25.0±0.5●	46.8±0.6●	97.0±0.2	42.3±1.9●	80.9±1.5●	77.8±0.7●	5/2/0
MLP+Mean	40%	87.3±0.8●	29.6±0.6●	60.8±0.6●	97.0±0.2	**47.2±2.1**	80.6±2.2●	87.4±0.6●	5/2/0
MLP+KNN[10]		86.9±0.7●	25.3±0.7●	51.5±0.8●	96.6±0.2●	44.1±1.6●	79.6±2.2●	80.9±0.5●	7/0/0
MLP+MICE[2]		**91.3±0.7**○	29.9±0.6●	61.5±0.7●	97.1±0.2	45.7±1.8	85.6±1.3●	**90.8±0.5**○	3/2/2
RF+Mean		89.3±0.9○	31.2±0.7●	68.5±0.5	**97.4±0.2**○	45.6±1.8●	**89.8±1.1**○	89.4±0.4●	3/1/3
RF+KNN[10]		88.8±0.6	26.5±0.5●	58.6±0.6●	96.9±0.2●	43.5±2.2●	85.1±1.4●	81.7±0.6●	6/1/0
RF+MICE[2]		90.7±1.0○	31.8±0.8●	65.5±0.7●	97.2±0.2	45.3±2.1●	89.2±1.0○	89.2±0.5●	4/1/2
RSS		80.6±0.9	**23.7±0.6**	**48.1±0.4**	**96.6±0.2**	**40.8±2.7**	76.0±1.4	**76.1±0.9**	
LR+Mean		78.5±0.8●	21.8±0.6●	29.5±0.5●	95.5±0.3●	37.4±1.8●	63.2±1.1●	59.2±0.6●	7/0/0
LR+KNN[10]		73.9±1.3●	20.4±0.5●	23.6±0.8●	95.2±0.2●	35.1±1.6●	49.8±2.3●	51.5±0.8●	7/0/0
LR+MICE[2]		80.2±1.0●	21.7±0.5●	31.4±0.5●	95.9±0.2●	38.2±1.5●	69.1±1.9●	63.4±0.9●	7/0/0
MLP+Mean	60%	78.8±1.0●	21.6±0.2●	40.3±0.5●	96.0±0.3●	40.4±2.2	70.5±2.2●	72.5±0.5●	6/1/0
MLP+KNN[10]		73.1±1.0●	19.2±0.6●	27.6±0.7●	95.4±0.3●	36.4±2.3●	54.9±2.6●	58.4±0.6●	7/0/0
MLP+MICE[2]		81.5±0.8	21.7±0.5●	40.7±0.7●	96.1±0.2●	40.7±1.7	74.4±1.5●	74.9±1.2●	5/2/0
RF+Mean		81.3±1.1	22.8±0.4●	47.4±0.5●	96.2±0.3●	37.7±1.6●	**82.0±1.7**○	75.0±0.6●	5/1/1
RF+KNN[10]		77.2±1.0●	18.6±0.4●	35.6±0.4●	95.5±0.3●	35.8±2.6●	65.0±1.9●	60.8±0.9●	7/0/0
RF+MICE[2]		**82.0±1.0**○	23.1±0.2●	44.6±1.0●	96.1±0.2●	37.7±1.1●	78.3±1.4○	74.0±0.8●	5/0/2
RSS		75.3±1.2	**21.0±0.6**	**35.8±0.7**	**95.5±0.3**	**39.2±1.6**	66.2±1.4	**65.0±0.8**	
LR+Mean		75.3±1.2	20.4±0.3●	22.9±0.3●	94.6±0.2●	36.2±2.0●	56.4±2.0●	51.0±0.8●	6/1/0
LR+KNN[10]		69.8±1.6●	18.9±0.4●	17.7±0.5●	94.6±0.3●	34.2±3.0●	43.1±1.6●	43.5±1.2●	7/0/0
LR+MICE[2]		75.3±1.0	20.2±0.4●	23.3±0.5●	94.7±0.2●	36.1±2.0●	59.8±2.9●	52.4±1.1●	6/1/0
MLP+Mean	70%	73.5±1.3●	19.1±0.5●	30.3±0.5●	94.7±0.3●	39.1±1.0	62.6±1.7●	61.1±0.8	5/2/0
MLP+KNN[10]		68.3±1.7●	16.7±0.4●	19.7±0.5●	94.7±0.2●	35.9±2.8●	46.5±1.7●	47.9±0.7●	7/0/0
MLP+MICE[2]		74.3±0.7●	18.7±0.8●	30.0±0.4●	95.1±0.2●	38.0±1.9●	64.6±2.1●	61.9±1.2●	7/0/0
RF+Mean		75.8±1.0	19.0±0.3●	34.5±0.6●	95.2±0.2●	33.7±2.2●	**73.8±1.0**○	64.3±0.9●	5/1/1
RF+KNN[10]		71.3±2.3●	16.2±0.4●	26.8±0.5●	94.8±0.2●	34.0±1.1●	56.3±1.9●	51.0±0.8●	7/0/0
RF+MICE[2]		**76.2±0.8**	19.8±0.5●	32.7±0.7●	95.0±0.1●	35.0±1.3●	69.6±1.9○	61.6±0.7●	5/1/1

Note: We report the average accuracy and standard deviation of 10 trials. ●/○ indicates that our RSS is significantly better/worse than the corresponding method (pairwise t-tests at 95% significance level).

From Table 9 we can get the following observations:

1) Multiple-imputation method (MICE) achieves better performance than single-imputation methods (Mean and KNN). However, the computing resources are too expensive for MICE, especially in the training process. For example, on the 357-dimensional dataset protein, MICE takes about one day to finish the imputation process and it indicates that it is not practical for many real-world applications with even higher dimensionalities. In contrast, single-imputation methods are efficient for training and inference but they are not accurate enough.

2) Our method RSS has higher accuracies but slower running speed than single-imputation methods. When compared with MICE, our method RSS has higher accuracies in most cases. More importantly, the training cost has been reduced a lot in contrast to MICE, especially on the relatively high dimensional datasets dna and protein.

3) With the increase of feature dimensionality, the training time for KNN, MICE and RSS also increases. However, the training overhead of MICE increases the most as the dimensionality increases among these methods, which corresponds to the previous time

Table 9. Speed Comparisons Between RSS and Other Methods

Methods	PMV	Yeast			Dna			Protein		
		Train (s)	Infer (s)	Acc. (%)	Train (s)	Infer (s)	Acc. (%)	Train (s)	Infer (s)	Acc. (%)
RSS	20%	69.15	0.83	52.7±1.9	260.47	6.92	**89.2±0.9**	2474.18	107.89	**63.9±0.4**
Mean		6.09	0.01	53.0±2.6	3.98	0.02	88.2±0.7	14.13	0.34	62.5±0.4
KNN[10]		7.27	0.54	51.1±1.7	6.37	2.05	88.2±0.9	347.47	150.88	61.4±0.7
MICE[2]		6.16	0.02	**54.7±1.9**	13973.27	21.88	89.1±0.8	39754.31	24.67	62.3±0.5
RSS	60%	99.19	1.21	**40.8±2.7**	1061.29	29.24	**75.8±0.8**	3954.64	170.98	**56.1±0.6**
Mean		7.11	0.01	40.3±1.9	3.73	0.02	73.7±1.1	12.08	0.29	53.9±0.5
KNN[10]		7.66	0.65	36.6±1.9	69.5	1.4	69.5±1.4	491.53	210.66	48.3±0.5
MICE[2]		6.74	0.03	40.2±2.2	18343.63	36.47	75.6±1.3	82562.93	118.68	55.7±0.6

Note: 'Train (s)' and 'Infer (s)' denote the average training and inference time on each dataset, in seconds. 'Acc. (%)' denotes the average accuracy of 10 trials.

complexity analysis in Subsection 3.4.

4) In short, RSS has both higher accuracy and efficiency than multiple-imputation method (MICE). Furthermore, the current implementation of RSS has not been carefully optimized and thus it has the potential to be further accelerated.

5 Conclusions

In this paper, we proposed a random subspace sampling method RSS for classification with missing data. RSS enables us to use multiple values for each missing feature in different random subspaces to reflect better uncertainty, which is very effective in dealing with a large proportion of missing values. We showed that RSS is robust to different levels of missing data and is more efficient than multiple imputation methods such as MICE[2]. We conducted experiments on both incomplete and complete datasets under different levels of missing values. Experimental results show that RSS achieves superior performance than other comparison methods and the advantage will become larger with the increase of missing rate. In the future, we will further investigate our method from a theoretical perspective.

References

[1] García-Laencina P J, Sancho-Gómez J L, Figueiras-Vidal A R. Pattern classification with missing data: A review. *Neural Computing and Applications*, 2010, 19(2):263–282. DOI: 10.1007/s00521-009-0295-6.

[2] White I R, Royston P, Wood A M. Multiple imputation using chained equations: Issues and guidance for practice. *Statistics in medicine*, 2011, 30(4):377–399. DOI: 10.1002/sim.4067.

[3] Farhangfar A, Kurgan L A, Pedrycz W. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics*, 2007, 37(5):692–709. DOI: 10.1109/TSMCA.2007.902631.

[4] Juszczak P, Duin R P W. Combining one-class classifiers to classify missing data. In *Proceedings of the 5th International Workshop on Multiple Classifier Systems*, Jun. 2004, pp. 92–101. DOI: 10.1007/978-3-540-25966-4_9.

[5] Krause S, Polikar R. An ensemble of classifiers approach for the missing feature problem. In *Proceedings of the International Joint Conference on Neural Networks*, Jul. 2003, pp. 553–558. DOI: 10.1109/IJCNN.2003.1223406.

[6] Polikar R, DePasquale J, Mohammed H S, Brown G, Kuncheva L I. Learn^{++}.MF: A random subspace approach for the missing feature problem.

Pattern Recognition, 2010, 43(11):3817–3832. DOI: 10.1016/j.patcog.2010.05.028.

[7] Ghahramani Z, Jordan M I. Supervised learning from incomplete data via an EM approach. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, Nov. 1994, pp. 120–127.

[8] Ahmad S, Tresp V. Some solutions to the missing feature problem in vision. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, Nov. 1993, pp. 393–400.

[9] Quinlan J R. C4.5: Programs for machine learning. *Machine Learning*, 1994, 16(3):235–240. DOI: 10.1007/BF00993309.

[10] Batista G E, Monard M C. A study of k-nearest neighbour as an imputation method. *Hybrid Intelligent Systems*, 2002, 87(48):251–260. DOI: 10.1109/METRIC.2004.1357895.

[11] Schafer J L. Analysis of Incomplete Multivariate Data (1st edition). CRC Press, 1997.

[12] Zhao Y, Udell M. Missing value imputation for mixed data via gaussian copula. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2020, pp. 636–646. DOI: 10.1145/3394486.3403106.

[13] Rubin D B. Multiple Imputation for Nonresponse in Surveys (1st edition). John Wiley & Sons, 2004.

[14] Houari R, Bounceur A, Tari A K, Kecha M T. Handling missing data problems with sampling methods. In *Proceedings of the 7th International Conference on Advanced Networking Distributed Systems and Applications*, Jun. 2014, pp. 99–104. DOI: 10.1109/INDS.2014.25.

[15] Stekhoven D J, Bühlmann P. Missforest–nonparametric missing value imputation for mixed-type data. *Bioinformatics*, 2012, 28(1):112–118. DOI: 10.1093/bioinformatics/btr597.

[16] Zhou Z H. Ensemble Methods: Foundations and Algorithms (1st edition). CRC Press, 2012.

[17] Ho T K. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998, 20(8):832–844. DOI: 10.1109/34.709601.

[18] Breiman L. Random forests. *Machine learning*, 2001, 45(1):5–32. DOI: 10.1023/A:1010933404324.

[19] Sharpe P, Solly R. Dealing with missing values in neural network-based diagnostic systems. *Neural Computing & Applications*, 1995, 3(2):73–77. DOI: 10.1007/BF01421959.

[20] Jiang K, Chen H, Yuan S. Classification for incomplete data using classifier ensembles. In *Proceedings of the International Conference on Neural Networks and Brain*, Apr. 2005, pp. 559–563. DOI: 10.1109/ICNNB.2005.1614675.

[21] Cao Y H, Wu J, Wang H, Lasenby J. Neural random subspace. *Pattern Recognition*, 2021, 112:Article No. 107801. DOI: 10.1016/j.patcog.2020.107801.

[22] Little R J, Rubin D B. Statistical Analysis with Missing Data (3rd edition). John Wiley & Sons, 2019.

[23] Mazumder R, Hastie T, Tibshirani R. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 2010, 11(80):2287–2322.

[24] Huang S J, Xu M, Xie M K, Sugiyama M, Niu G, Chen S. Active feature acquisition with supervised matrix completion. In *Proceedings of the*

24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Jul. 2018, pp. 1571–1579. DOI: 10.1145/3219819.3220084.

[25] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, Jul. 2015, pp. 448–456.

[26] Kingma D P, Ba J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, May 2015.

Yun-Hao Cao is currently a Ph.D. candidate in the Department of Computer Science and Technology in Nanjing University, Nanjing. He received his BS degree in Computer Science and Technology from Nanjing University. His research interests are computer vision and machine learning.

Jian-Xin Wu is currently a professor in the School of Artificial Intelligence at Nanjing University, Nanjing, and is associated with the State Key Laboratory for Novel Software Technology, China. He received his BS and MS degrees from Nanjing University, and his PhD degree from the Georgia Institute of Technology, all in computer science. He has served as an (senior) area chair for CVPR, ICCV, ECCV, AAAI and IJCAI, and as an associate editor for the IEEE Transactions on Pattern Analysis and Machine Intelligence. His research interests are computer vision and machine learning.