Hindawi
Complexity
Volume 2024, Article ID 4737963, 15 pages
https://doi.org/10.1155/2024/4737963

Research Article

A Probabilistic Approach for Missing Data Imputation

Muhammed Nazmul Arefin [1] **and Abdul Kadar Muhammad Masum** [2]

[1]*Department of Computer Science and Engineering, International Islamic University Chittagong, Chattogram 4318, Bangladesh*
[2]*Department of Software Engineering, Daffodil International University, Dhaka 1216, Savar, Bangladesh*

Correspondence should be addressed to Muhammed Nazmul Arefin; nazmul.arefin@iiuc.ac.bd

Received 21 February 2023; Revised 10 November 2023; Accepted 3 January 2024; Published 19 January 2024

Academic Editor: Hassan Zargarzadeh

In the context of data analysis, missing data imputation is a vital issue due to the typically large scale and complexity of the datasets. It often results in a higher incidence of missing data. So, addressing missing data through the imputation technique is essential to ensure the integrity and completeness of the data. It will ultimately improve the accuracy and validity of the data analysis. The prime objective of this study is to propose an imputation model. This paper presents a method for imputing missing employee data through a combination of features and probability calculations. The study utilized employee datasets that were collected from the Kaggle along with primary data collected from RMG factories located in Chittagong. The suggested algorithm demonstrated a notable level of accuracy on the datasets, and the average accuracy for each identified technique was also quite satisfactory. This study contributes to the existing body of research on missing data imputation in big data analysis and offers practical implications for handling missing data in different datasets. Usage of this technique will enhance the accuracy of data analysis and decision-making in organizations.

1. Introduction

Missing data refer to the absence of values in one or more attributes within the dataset. The reasons for missing data can vary, including clerical errors during data entry, system malfunctions, voluntary employee nondisclosure, or data corruption during storage and transmission. Regardless of the underlying cause, missing data can lead to significant issues and hinder the organization's ability to draw accurate and reliable conclusions from the dataset. In today's data-driven business landscape, organizations heavily rely on employee datasets to gain valuable insights into their workforce, optimize operations, and make informed decisions. These datasets encompass a wide range of information, including employee demographics, performance metrics, job roles, salary details, attendance records, and other essential attributes. However, one common challenge that plagues these datasets is the presence of missing data. To forecast results from massive datasets, several machine learning and data mining methods are frequently utilized. These algorithms often produce accurate predictions unless the data used to train them are incorrect. One of the most important parts of the data analysis is data purification also known as data preprocessing. Also, missing data handling is one of the significant parts of it. Without proper preprocessing, machine learning algorithms are not able to perform excellent predictions which creates a situation where decisions are frequently erroneous. Hence, missing values are one of the biggest problems with data integrity which creates a big issue for a good data analysis. A dataset with missing values may result in skewed results and drastically higher processing costs [1]. Avoiding missing records can be an easy resolution for a dataset with fewer data. On the other hand, deleting missing entries could result in a large information loss for larger datasets [2]. Therefore, trustworthy imputation methods are required to address this problem. Many statistical techniques, such as replacing by mean, median, or mode, are frequently employed to handle missing values in datasets. However, for the employee dataset, we need an appropriate missing data handling technique. In this paper, we introduced combinational probabilistic analysis, a novel method for imputation of

missing data. Additionally, we implemented a number of well-known single imputation, multiple imputations, and machine learning-based imputation techniques. We have performed comparisons with all of the implemented approaches with the proposed algorithm in terms of accuracy and $f1$ score. In contrast to previous methods, the comparison demonstrates better results for the suggested method.

The second section of the paper has literature reviews. The third section contains the proposed approach. The fourth and fifth sections present the result analysis and discussion. Also, finally, the conclusion part, along with the contribution, is in the sixth section.

2. Literature Review

In this section, we have presented a brief description of different types of missing data as long as the necessary prior work and literature. In any dataset, two types of missing can occur: variable missing and value missing [3]. Missing variable is the worst case for a dataset if the variable plays a vital role in the analysis. Mr. Nikolas mentioned two ways to handle missing variables in [4]. First is precluding the entire variable. The second is inferring missing variables from any other dataset where relevant variables exist, such as in the current dataset. This method requires more investigation before inference, since, in many circumstances, imputing from another dataset will not provide the correct value for the missing variable. To investigate, the calculation is required to find whether there is any bias in the case of omitted variables or in the case of imputed variables. If the bias created for excluding missing variables in the dataset is larger than the bias created for imputation, then imputation is considered as the improvement [4].

In terms of "value missing," there are three kinds in most cases [5]. First is the missing completely at random (MCAR), then, missing at random (MAR), and finally, missing not at random (MNAR). Before we go into the specifics of those types, it is important to understand the terms "observed data" and "unobserved data." Observed data are data of observed variables. The observed variable is one that can be measured in the dataset, e.g., elements such as survey replies, performance assessment scales, duration on task, and job completion, and unobserved variables or latent variables, on the other hand, are those that cannot be assessed directly in the dataset, or even exist. Although latent variables cannot be measured directly, they can be indirectly measured by employing seen ones. For instance, an unobserved variable of intelligence lies behind a couple of observed variables such as performance and presentation.

When data missing is not dependent on observed and unobserved data, it is called missing completely at random (MCAR) [5]. This form of missing data does not introduce bias into the dataset, but it does reduce the number of populations that can be analyzed. As this is independent of observed and unobserved, there is less chance to get any pattern to use for further imputation. For example, if a survey result has 6% missing responses arbitrarily, it is MCAR. In the case of missing at random (MAR), other variables in the dataset have the ability to impute the missing values. In MAR, missing data are related to the observed but not the unobserved data. For instance, in a survey of "the intention for leaving my current job," male employees participated less than females would be MAR. Another explanation for being MAR is that the survey is more focused on gender (observed variable) than on intention (latent variable). If the dataset's only complete data are used for analysis while there exists MAR in the dataset, then the result will not be biased [6].

When missing data are related to unobserved variables, then it is called missing not at random (MNAR), i.e., those missing values are not directly related to any measurable variables. As an illustration, some respondents refused to take part in a survey that was designed to investigate a nonmeasurable fact, e.g., cigarette addiction (hidden variable). So, as many participants who are already addicted refused to participate in the survey, this is MNAR. The reason for the missing data in this case is well recognized, but imputing that data is not that simple. Having MNAR in the dataset, inference by only complete data can bring biased results.

In the domain of missing data imputation, numerous research studies have been documented. In this context, we are compiling a selection of significant works and summarizing their key findings in Table 1.

After studying them, we found that there are some algorithms that are less sensitive to the interconnectedness of variables in an employee dataset. Imputing missing values in isolation [21] can break the relationships between variables and lead to inconsistent or implausible imputed data.

(i) Employee data typically include a variety of variables, such as age, gender, job title, salary, performance rating, years of experience, educational qualifications, training history, and skills and competencies. These variables are often interconnected, and imputing missing values for one variable in isolation can lead to inconsistent or implausible imputed values for other variables. For example, suppose we have a dataset of employee data with a missing value for an employee's salary. If we impute the missing value for salary in isolation, we may end up with an imputed value that is inconsistent with the employee's other characteristics, such as their job title, experience, and performance rating. For example, we may impute a high salary for an employee with a low-level job title and a low-performance rating. This is unlikely to be a realistic scenario.

(ii) Another problem with imputing missing values in isolation is that it can lead to implausible imputed data. For example, we may impute a value for an employee's age that is outside of the possible range of ages for humans.

While substantial research has been conducted on missing data imputation techniques in various domains,

Table 1: Literature review of ML algorithms and highlights.

ML category	Method	Highlights
Clustering	Best fit missing value imputation	Used for the IoT datasets. The paper provides a comparison of BFMVI with other existing algorithms and shows that BFMVI outperforms them in terms of accuracy and efficiency [7]
	Cluster-directed framework for neighbor-based imputation	A brand-new cluster-directed framework is suggested by the authors. CFNI: cluster-directed framework for neighbor-based imputation, which uses data clustering alone to lead the identification of closest neighbors in order to get a more precise imputed value [8]
	C-means	Used to impute the value in missing places by similar entries in the complete datasets [9]. Used in distributed datasets [10]
	K-means	Patil et al. used it to impute missing value in their work [11]
Deep learning	Deep neural network	Able to fit the data closely, and can accurately predict new data points [12]
	Long-short-term memory	Demonstrates good performance for time series missing values [13]
Ensemble	AdaBoost	In [14], authors showed that the method is good enough to resilient missing data to identify hemodynamic instability in ICU patients early on
	eXtreme gradient boosting	Employs feature selection and superior accuracy [15, 16]
	Random forest	In [17], authors used random forest to estimate categories for similarity measuring to impute missing data
Neural network	Multilayer perceptron (MLP)	In [18], authors showed the good results in categorical variables using MLP
Instance based	k-nearest neighbors (kNN)	Pan et al. [19] considered the feature relevance which was measured by their modified KNN
	Support vector machine (SVM)	It was used to impute missing data for activity-based transportation model [20]

there exists a notable research gap concerning the specific challenges and effective solutions for handling missing data in employee datasets. Although organizations increasingly rely on employee datasets to drive data-driven decision-making and enhance human resource management, limited research has been dedicated to comprehensively addressing missing data issues in this context.

Employee datasets possess unique characteristics, including diverse employee attributes, hierarchical structures, and temporal dependencies. Existing missing data imputation techniques often treat missing values generically, without considering the specific nature of employee-related attributes. There is a need for research that tailors imputation methods to the characteristics of employee datasets, ensuring more accurate and context-aware results.

3. Methodology

We are introducing a method called combination probabilistic analysis, where, by creating combinations of important features existing in the dataset, selecting which class of target attribute gives the highest probability for each node of those combinations. In addition, the results obtained with the methods used in various researches have also been analyzed.

Throughout this section, we explicitly define our problem domain and detail each step in the process of achieving the study goal.

3.1. Dataset Description. For our experiments, we employed four datasets as shown in Table 2. One of them is the employee dataset, which was obtained from a handful of RMG factories in Chittagong. Another is the Kaggle employee-attrition dataset [22]. In the primary dataset around 1500 employees, data under 20 attributes were extracted. Many of those attributes are categorical, and some of them are numerical. The dataset from Kaggle has similar properties. The Kaggle dataset has 1050 training data and 442 testing data in separate files within 34 attributes. Also, the IBM employee dataset [23] has 1470 employee records with 35 attributes. All of the datasets contain categorical key attributes, for instance, designation, department, and employee type. Some of the categorical attributes are ordinal, and some of them are nominal. So, we have decided to encode ordinal attributes into ordered numerical attributes and nominal attributes into dummy encoded attributes. Some other numerical attributes such as salary and age are encoded using intervals as those values are scattered. We also used the hair-eye color dataset [24] to assess how well the proposed imputation technique functions.

3.2. Data Preprocessing. Each of the datasets contains categorical key attributes, for instance, designation, department, and employee type. Some of the categorical attributes are ordinal, and some of them are nominal. So, we have encoded ordinal attributes into ordered numerical attributes and nominal attributes into dummy encoded attributes. Some other numerical attributes such as salary and age are encoded using intervals as those values are scattered. We encoded "AgeLevel" by classifying it as "Children," "Young," "Adult," and "Senior," as per the National Statistical Office of Canada's age classifications [25].

Table 2: Dataset description.

Datasets	#Records	#Attributes
Local employee dataset	1500	20
Employee-attrition dataset [22]	1492	34
IBM HR analytics attrition dataset [23]	1470	35
Hair-eye color dataset [24]	592	5

In forming a model, the procedure of feature selection includes minimizing the number of attributes. In certain situations, dropping the number of input variables might enhance the proficiency of the model while also decreasing the computing cost of modeling. In this work, we selected significant features from a large set of attributes using a variety of feature selection techniques. The test statistic used to determine the relationship or statistical link between two continuous variables is called Pearson's correlation coefficient [26]. We utilized the scikit-learn scoring function named "r_regression," that can calculate Pearson's r for every attribute and the target. Another popular feature selection approach is univariate feature selection. In this method, the most effective attributes are preferred using univariate statistical tests. We disregard the other characteristics while examining the link between a single feature and the target variable. That is, why it is referred to as "univariate." In this approach, scikit-learn provides a function called "SelectKBest," that eliminates all features except for the top k scoring ones. In this study, we utilized "SelectKBest" function.

3.3. Multiple Imputations

3.3.1. Multiple Imputations by Chained Equations (MICE). MICE is a statistical method used for handling missing data by imputing (filling in) missing values with plausible estimates [27]. MICE is a flexible imputation method that iteratively imputes missing values in a dataset multiple times to generate multiple complete datasets with imputed values. These datasets can then be used for subsequent analyses. The key idea behind MICE is to impute missing data for each variable by modeling it as a function of other variables with observed values. This is achieved through a series of conditional imputations, often using regression models.

3.3.2. Mathematical Representation

(i) Dataset: consider a dataset with n observations and p variables. The dataset can be represented as a matrix Y, where each row i represents an observation and each column j represents a variable:

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \end{bmatrix}. \quad (1)$$

(ii) Missing data indicators: we introduce binary indicator variables $(R_i j)$ to indicate whether a value is observed $(R_i j = 1)$ or missing $(R_{ij} = 0)$:

$$R_{ij} = \begin{cases} 1, & \text{if } y_{ij} \text{ is observed,} \\ 0, & \text{if } y_{ij} \text{ is missing.} \end{cases} \quad (2)$$

(iii) Imputation process: the MICE algorithm imputes missing values for each variable j in an iterative manner. For variable j, we model its missing values (Y_{ij}) as a function of other variables (Y_{-j}) in the dataset. This is carried out using regression models, and the imputed values are denoted as $\widehat{Y}_{ij}$. The imputation model can be represented as follows:

$$Y_{ij} = f_j\left(Y_{-j}\right) + \epsilon_{ij}, \quad (3)$$

where Y_{ij} is the missing value to be imputed, f_j is a regression model for variable j, Y_{-j} represents all other variables except j, and ϵ_{ij} represents the error term.

(iv) Multiple imputations. The MICE process is repeated multiple times (usually m times) to create m complete datasets with imputed values. This generates a set of imputed datasets $Y^{(1)}, Y^{(2)}, \ldots, Y^{(m)}$.

3.4. Machine Learning-Based Imputation. In the evaluation of machine learning algorithms, several prominent approaches were explored. These approaches fall into distinct categories, each offering its own set of advantages and characteristics. First, decision tree-based classifiers were investigated, providing a transparent and interpretable way to make predictions. Next, deep learning-based approaches, specifically artificial neural networks, and multilayer perceptrons were applied to the datasets, harnessing the power of complex neural architectures for pattern recognition. Ensemble methods, including AdaBoost and eXtreme gradient boosting, were also considered, which combine multiple models to enhance predictive performance. Finally, in the instance-based approach, K-nearest neighbors (KNN) and support vector machine (SVM) classifiers were tested, relying on the proximity of data points to make predictions. Each of these algorithmic categories was thoroughly assessed to determine their effectiveness in solving the given problem.

3.4.1. Decision Tree Classifier (DTC). DTC is a popular supervised learning algorithm used for classification and regression tasks. It is a nonparametric model that predicts the value of a target variable based on a set of input features [28]. The algorithm works by recursively partitioning the data into subsets based on the values of the input features, until each subset contains only instances of a single class or has reached a maximum depth or minimum number of instances.

In decision tree classification, one common criterion used for making decisions at each node is Gini impurity. Gini impurity measures the impurity or disorder in a dataset. For a binary classification problem (two classes, typically denoted as 0 and 1), the Gini impurity for a node is calculated as follows:

$$\text{GiniImpurity} = 1 - \left(\frac{p_0}{p_0 + p_1}\right)^2 - \left(\frac{p_1}{p_0 + p_1}\right)^2, \quad (4)$$

where (i) p_0 is the proportion of samples in the node belonging to class 0 and (ii) p_1 is the proportion of samples in the node belonging to class 1.

Now, let us use Gini impurity to understand how a decision tree is built.

(i) Step 1: Root node selection

(1) At the root node, we have the entire dataset, and we consider all available features.
(2) For each feature, we calculate the Gini impurity for different possible splits (using different thresholds). The feature and threshold that result in the lowest Gini impurity are chosen. This split maximizes class separation at the root node.

(ii) Step 2: Recursive splitting

(1) After selecting the root split, we repeat the process for each child node.
(2) We consider the subset of data that fall into a node and evaluate Gini impurity for different feature thresholds. The best split is chosen to minimize Gini impurity.
(3) This recursive process continues until we reach a stopping condition, such as reaching a maximum tree depth or having a minimum number of samples per leaf.

(iii) Step 3: Leaf node assignment

(1) Once the splitting is complete, the leaf nodes represent subsets of data with relatively pure class labels
(2) The predicted class for each leaf node is determined based on the majority class of the samples in that node

When we want to classify a new data point using the decision tree:

(i) We start at the root node and evaluate the feature value of the data point.
(ii) We follow the path down the tree by comparing the feature value to the threshold in each internal node, as determined during the tree-building process.

(iii) This process continues until we reach a leaf node. The class label associated with that leaf node becomes the predicted class for the new data point.

3.4.2. Artificial Neural Network (ANN). ANN [29] is a type of machine learning algorithm inspired by the structure and function of the human brain. It consists of a network of artificial neurons that are interconnected to process information and make predictions. An ANN typically consists of three types of layers: input layer, hidden layer (s), and output layer. Each layer contains a certain number of neurons, which are responsible for processing the input data, transforming it through a set of weights and biases, and generating an output. During training, the ANN learns by adjusting the weights and biases of the neurons to minimize the difference between the predicted output and the actual output. This process is achieved through backpropagation, which involves calculating the error of the predicted output and propagating it back through the layers to adjust the weights and biases. ANNs can be used for various tasks, such as classification, regression, and clustering, and have achieved remarkable success in fields such as computer vision, speech recognition, and natural language processing. They are known for their ability to learn from complex and large datasets and generalize well to new data. Let us break down ANNs into some intuitive components:

(1) Neurons (artificial neurons): in an ANN, the basic building blocks are artificial neurons. These neurons are similar to the neurons in our brain, which receive signals, process them, and produce an output. Each artificial neuron takes multiple inputs, processes them, and produces an output. Think of these neurons as tiny decision-makers.

(2) Layers: neurons are organized into layers. ANNs typically consist of an input layer, one or more hidden layers, and an output layer. Information flows from the input layer through the hidden layers to the output layer. Each layer contains a set of neurons that perform specific tasks.

(3) Weights and connections: the strength of the connections between neurons is represented by weights. These weights determine how much importance each input has on the neuron's decision. Weights are adjusted during the learning process, allowing the network to adapt and improve its performance. Activation Function. Neurons use an activation function to determine their output. This function introduces nonlinearity into the network, enabling it to model complex relationships in the data. Common activation functions include the sigmoid, ReLU (rectified linear unit), and tanh (hyperbolic tangent).

(4) Learning and training: ANNs learn from data through a process called training. During training, the network is presented with input data along with the correct or expected output (supervised learning). The network makes predictions, and the difference between its predictions and the correct output is measured using a loss function. The network then adjusts its weights to minimize this loss, making it better at making predictions.

(5) Backpropagation: the mechanism by which ANNs adjust their weights during training is called backpropagation. It is like the network learning from its mistakes. If the prediction is wrong, the network "learns" how to change its weights to improve future predictions.

(6) Deep learning: when ANNs have multiple hidden layers (deep neural networks), they become capable of learning intricate, hierarchical patterns in data. Deep learning has been revolutionary in tasks such as image recognition, natural language processing, and game playing.

(7) Applications: ANNs find applications in a wide range of fields, from image and speech recognition to autonomous vehicles, medical diagnosis, recommendation systems, and more. They excel in tasks where patterns or relationships are complex and difficult to express with traditional programming.

In essence, ANNs are mathematical models that can automatically learn and adapt to solve complex problems by processing data through interconnected artificial neurons. They have proven to be incredibly versatile and powerful, leading to significant advancements in various areas of technology and science.

3.4.3. Ensemble Learning-Based Imputation

(1) AdaBoost, short for adaptive boosting [30], is a popular ensemble learning algorithm used in machine learning. It works by combining multiple weak classifiers (classifiers that perform only slightly better than random guessing) into a strong classifier that can make accurate predictions. The AdaBoost algorithm works as follows:

   (a) Initialization: each instance in the training dataset is assigned an equal weight.
   (b) Training weak classifiers: a weak classifier is trained on the training data, and its performance is evaluated. The instances that are misclassified by the weak classifier are assigned a higher weight to give them more importance in subsequent iterations.
   (c) Combining weak classifiers: the weak classifiers are combined into a strong classifier by assigning a weight to each weak classifier based on its performance. The better a weak classifier performs, the higher its weight will be in the final ensemble.
   (d) Final classification: the ensemble of weak classifiers is used to classify new instances. Each weak classifier votes on the classification of the instance, and the final classification is determined by the weighted sum of the votes.

AdaBoost is known for its ability to improve the accuracy of weak classifiers and its resistance to overfitting. It has been successfully applied to a wide range of problems, including face detection, speech recognition, and text classification. However, it can be sensitive to noisy data and outliers and may require careful tuning of hyperparameters.

(2) eXtreme gradient boosting (XGBoost) [31] is a powerful ensemble learning algorithm that uses a gradient boosting framework to produce a highly accurate prediction model. XGBoost was developed to overcome some of the limitations of traditional gradient boosting methods, such as overfitting and slow training times. The XGBoost algorithm works by building an ensemble of decision trees, where each tree is trained to correct the errors of the previous tree. It uses a combination of regularization techniques, such as L1 and L2 regularization, and early stopping to prevent overfitting and improve generalization performance. XGBoost also uses a novel optimization technique called gradient-based one-side sampling (GOSS) to speed up training by selecting a subset of instances for each tree. The key features of XGBoost are as follows:

(a) Regularization: XGBoost uses L1 and L2 regularization to control overfitting and improve model generalization
(b) Tree pruning: XGBoost uses early stopping and maximum depth constraints to prune the decision trees, which helps to prevent overfitting
(c) Ensemble learning: XGBoost combines multiple decision trees to create a more accurate prediction model
(d) Parallel processing: XGBoost can use parallel processing to speed up the training process

XGBoost is widely used in a variety of applications, including classification, regression, and ranking problems. It has won numerous machine learning competitions and is considered one of the best-performing machine learning algorithms available today.

For doing the above-mentioned experiments, we have implemented each algorithm in Python. The primary dataset was collected from RMG factories, and the secondary two employee datasets were from Kaggle. A brief description of those datasets was provided in Section 3.1.

3.5. Proposed Method. In this study, a new approach has been introduced called combinational probabilistic analysis. It has the following steps:

(1) Combine attributes: create combinations of two or three attributes from the dataset. For example, if the dataset has attributes A, B, and C, combinations could be AB, AC, or BC.

(2) Create nodes: for each attribute combination XY, where *X* and Y are attributes, create a node with the class value of the attributes used in it. For example, if attribute A has classes 1 and 2, and attribute B has classes 1, 2, and 3, then the nodes for the AB 7 combination would be 11, 12, 13, 21, 22, 23.

(3) Calculate probability: calculate the probability $P(C_k)$, where C_k is the class value of the target attribute, for each node XY_{ij}. Also, record the number of records $N(XY_{ij})$ against that node.

(4) Select class:

(a) Select the class C_k for a node XY_{ij} based on the highest probability
(b) If the probability of any two target attribute classes C_a and C_b for a node XY_{ij} is close or equal ($|P(C_a) - P(C_b)| \leq \epsilon$) and the number of records $N(XY_{ij})$ against that node is not extremely low ($N(XY_{ij}) > \delta$), classify that node using nodes of a new combination to ensure that significant records are not omitted
(c) If the probability of both classes is equal ($|P(C_a) = P(C_b)|$) and the number of records $N(XY_{ij})$ against that node is negligible, select the class C_k with the higher probability of the target attribute

In this way, the target classes that have been set for all the nodes in each combination must be stored, which will be used to find the missing value later. Figure 1shows the flowchart of the proposed method. Algorithm 1 refers to the algorithm for combining features, and Algorithm 2 refers to the algorithm for calculating probabilities.

3.5.1. Illustration. Figure 2 shows attributes A, B, and C and a target attribute T. Each node in the diagram represents a combination of attributes A and B and shows the probability of the target class for that node. If the probability of a target class is higher for a node, that class is chosen for that node. If the probabilities are equal, the node is reclassified using the AC combination.

For example, in node 11, the chance of target class 1 is higher, so class 1 is chosen for that node. However, in node 12, the probabilities of target classes 1 and 2 are equal, so the node is reclassified using the AC combination. Node 12 of AB combination exists in 2 nodes of AC combination, i.e., 11 and 12. The probability of target classes 1 and 2 is found for both 11 and 12. Within the AC combination, node 11 has a higher probability of target class 1. Therefore, if AB and AC's nodes are 12 and 11, class 1 can be chosen. And, if AB and AC's nodes are 12 and 12, respectively, then class 2 can be selected.

In node 23 of the AB combination, the probability of class 1 is 33% and the probability of class 2 is 66%. Normally, class 2 should be selected for this node, but node 23 appears 6 times in the dataset, which is 80% of the total records. So, the node needs to be reclassified. To do this, we find the nodes available for 23 of AB in the AC combination. As can be seen for nodes 21 and 22 of AC, the probability of class 2 is higher, so they are selected. All these results are stored in the following 2 separate tables. Table 3 is for target class one, and Table 4 is for target class two.

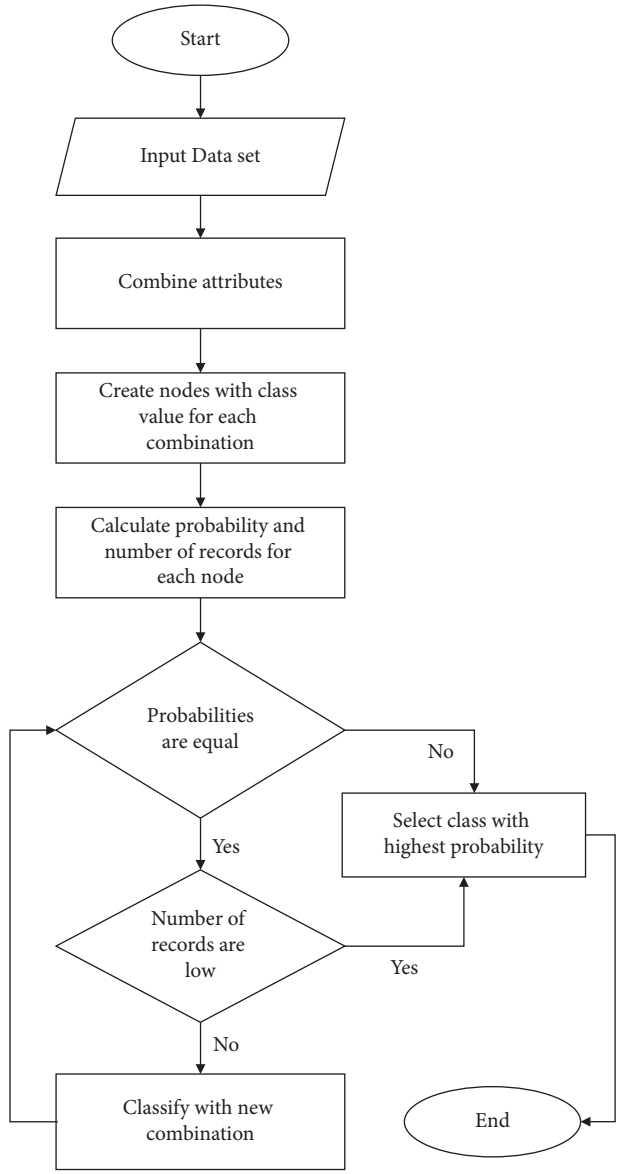

Figure 1: Flowchart of proposed model.

```
(1) function COMBINATION (attribute_x, attribute_y)
(2)                                   ▷ This function creates combinations
(3)     Pass In: attribute_x, attribute_y
(4)                ▷ Here, attributes are selected features from the dataset
(5)     combinations = empty list
(6)     for each class i of attribute_x do
(7)       for each class j of attribute_y do
(8)         combination_ij = (i, j)
(9)         add combination_ij to combinations list
(10)      end for
(11)    end for
(12)    Pass Out: combinations
(13) end function
```

Algorithm 1: Combination.

```
(1) function PROBABILITIES (dataset, combinations)
(2)       ▷ This function calculates the probability of a combination within the dataset
(3)     Pass In: dataset, combinations
(4)     probabilities = empty dictionary
(5)     for each class i of target attribute do
(6)       for each combination j in combinations do
(7)         count_combination_i = 0
(8)         count_target_i = 0
(9)         for each row r in dataset do
(10)          if r has combination j then
(11)            count_combination_i = count_combination_i + 1
(12)            if r has target class value i then
(13)              count_target_i = count_target_i + 1
(14)            end if
(15)          end if
(16)        end for
(17)        probability_i j_i = count_target_i/count_combination_i
(18)        probabilities[(i, j)] = probability_i j_i
(19)      end for
(20)    end for
(21)    Pass Out: probabilities
(22) end function
```

Algorithm 2: Probability calculation.

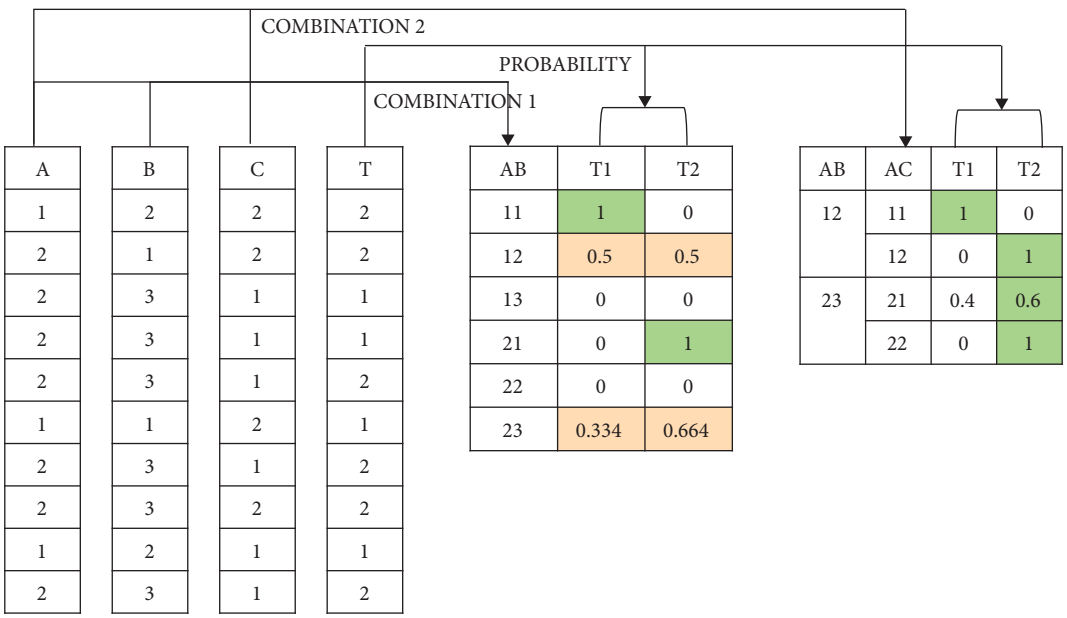

Figure 2: Illustration of combinational probability of attributes.

Table 3: Target class 1 prediction table.

AB	AC	Decision
11		1
12	11	1

Table 4: Target class 2 prediction table.

AB	AC	Decision
21	—	2
12	12	2
23	21	2
23	22	2

Based on the data of these three attributes A, B, and C for any missing data of T attribute of this dataset, we can make a missing imputation. For example, we have to impute the missing data in Table 5.

Here, we get node 12 in the first combination AB. As we can see from Table 3, no classification is carried out based on node 12 of the AB combination. This node is again divided by

Table 5: Sample missing data table.

SL	A	B	C	Missing
1	1	2	1	—
2	2	1	2	—
3	1	2	2	—

Table 6: Sample table after imputation.

SL	A	B	C	Imputed
1	1	2	1	1
2	2	1	2	2
3	1	2	2	2

AC combination. In the first row of test data, the AC combination has node 11. From prediction Table 4, we get target class 2 for nodes 12 (AB) to 11 (AC). Hence, for the first row, we impute 2 in the missing place of T. In the second row, the AB combination has node 21. From prediction Table 4, node 21 for AB combination has target class 2, so we impute the missing place using 2. In this way, we have Table 6 with the imputed values in the missing places of Table 5.

3.5.2. Justification of Combining Features and Performing Probability. The method here provided, which involves combining features and performing probability calculations for imputing missing values in an employee dataset, can be justified based on several key principles and considerations:

(i) Employee datasets are often rich and multifaceted, with various attributes that can influence one another. Combining features allows for the creation of composite attributes [32] that may capture complex relationships more accurately. By creating feature combinations, this method aims to represent the data distribution more effectively. Probability calculations then allow us to estimate the likelihood of a particular class or value for the target attribute based on the observed feature combinations.

For instance, the *job performance rating* likely depends on a combination of factors, including an employee's education, training, and experience. By combining these features (e.g., creating a composite feature like Education x Experience x Training), nuanced relationships can be captured that impact job performance. The imputation method leverages the information in these feature combinations to make more accurate predictions about an employee's job performance rating.

(ii) Many employee-related decisions require transparency and interpretability [33]. Feature combinations and probability calculations can lead to more interpretable imputations compared to black-box methods [34]. The method's use of conditional probabilities allows us to understand how imputed values are derived from observed data, making it easier to explain the imputation process to stakeholders.

For illustration, the use of feature combinations and probability calculations allows HR to explain that someone's high-performance rating is based on his education, training, and supervisor's assessment. As a result, the transparency in the imputation process fosters trust in HR decisions.

3.5.3. Time Complexity Analysis. The time complexity of the combination function is $O(n^2)$, where n is the number of classes in attribute_x. This is because the function needs to iterate over all possible combinations of the classes in attribute_x and attribute_y.

The time complexity of the probabilities function is $O(mn^2)$, where m is the number of classes in the target attribute. This is because the function needs to iterate over all possible combinations of the classes in attribute_x and attribute_y, and for each combination, it needs to iterate over all rows in the dataset.

The overall time complexity of the two functions is $O(mn^2)$, which is dominated by the time complexity of the probabilities function.

Here is a breakdown of the time complexity of each function:

(i) Combination function:

(1) Iterate over all classes in attribute_x: $O(n)$
(2) Iterate over all classes in attribute_y: $O(n)$
(3) Add each combination to the list of combinations: $O(1)$
(4) Total time complexity: $O(n^2)$

(ii) Probabilities function:

(1) Iterate over all classes in the target attribute: $O(m)$
(2) Iterate over all combinations of the classes in attribute_x and attribute_y: $O(n^2)$
(3) Iterate over all rows in the dataset: $O(n)$
(4) Calculate the probability of each combination: $O(1)$
(5) Add the probability to the dictionary: O(1)
(6) Total time complexity: $O(mn^2)$

The overall complexity of the two functions is $O(mn^2)$, which is dominated by the time complexity of the "probabilities" function. This means that the time it takes to run the two functions will be proportional to the product of the number of classes in the target attribute, m, and the square of the number of classes in attribute_x, n.

For example, if there are 10 classes in the target attribute and 5 classes in attribute_x, then the overall complexity of the two functions will be $O(10 * 5^2) = O(250)$. This means that it would take at least 250 steps to run the two functions.

The overall complexity can be reduced by using a more efficient algorithm to calculate the probabilities of the combinations. For example, we could use a hash table to store the probabilities of the combinations, so that we do not have to recalculate them for each row in the dataset. This would reduce the time complexity of the "probabilities"

function to $O(mn)$, which is still quadratic, but it would be a significant improvement over the original $O(mn^2)$ complexity.

3.6. Model Evaluation. We obtained accuracy and $F1$ scores to assess performance for all of the algorithms across all datasets. The basic categorization metric is accuracy. It only quantifies the proportion of accurate predictions a machine learning model has produced. Accuracy, however, is a weak statistic when dealing with imbalanced data since it cannot differentiate between different sorts of errors. Due to the ability of precision and recall to take into consideration the different types of errors that the model can produce, they are performance metrics that are better suited when the data are unbalanced. Precision and recall are combined into a single metric called the F1 score [35]. Having just one performance statistic instead of several is often considerably more practical. For the $F1$ score, the precision and recall mean should be calculated. It is sensible to choose the symmetrical average as they are both rates. Equation (5) shows the accuracy formula. Also, equation (6) shows the $F1$ score calculation formula.

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{number of total predictions}}, \tag{5}$$

$$\text{F1 score} = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{\text{precision} + \text{recall}}. \tag{6}$$

4. Result Analysis

In this section, we discuss the results found from the experiments we conducted. Several algorithms were implemented and tested. Among them, machine learning-based algorithms are available in different packages of Python including deep learning-based algorithms. We used datasets for imputation model testing purposes. We used "Gender" and "Age" attribute as target in all datasets. In "Gender" attribute, only binary (male/female) values were taken into consideration [25].

4.1. Single Imputation. In the first experimental setup, we found the following result as shown in Table 7. This table shows the result of different datasets for the single imputation method and the comparison with the proposed algorithm for "Gender" target attribute.

The table provides a comparison of different single imputation techniques, namely mean imputation and CPA imputation, on four datasets: "Local," "Kaggle," "IBM," and "HEC." The evaluation metrics used to assess the performance of these techniques are accuracy (Acc) and $F1$ score.

The mean imputation technique resulted in accuracies ranging from 0.40 to 0.50 across the datasets. It achieved relatively moderate $F1$ scores, ranging from 0.57 to 0.60. Mean imputation replaces missing values with the mean of the available data, providing a simple and straightforward approach.

Table 7: Result comparison with single imputation.

		Local	Kaggle	IBM	HEC
Mean	Acc	0.44	0.4	0.5	0.42
	$F1$ score	0.585	0.57	0.6	0
Proposed (CPA)	Acc	0.62	0.54	0.6	0.72
	$F1$ score	0.7	0.65	0.75	0.68

On the other hand, the CPA (combinational probability analysis) imputation technique showed higher accuracies, ranging from 0.54 to 0.72 across the datasets. It also yielded higher $F1$ scores, ranging from 0.65 to 0.75. CPA imputation takes into account the probabilities of missing values based on other variables, allowing for a more informed imputation process.

These results suggest that CPA imputation outperformed mean imputation in terms of both accuracy and $F1$ score (as shown in Figure 3). However, it is important to note that the choice of imputation technique may depend on the specific characteristics of the dataset and the nature of the missing values. Further analysis and experimentation may be required to determine the most appropriate imputation technique for a given scenario.

4.2. Multiple Imputations. The comparative result of the MICE and the proposed algorithm is given below in Table 8 for "Gender" target attribute in all datasets.

Figure 4 illustrates the comparative results of two imputation methods, MICE and CPA, on four different datasets (Local, Kaggle, IBM, and HEC). The evaluation metrics used are accuracy (Acc) and $F1$ score. Here is the result analysis:

(i) Mice imputation:

(1) Accuracy: MICE achieves an average accuracy of 0.49 across all datasets, with the highest accuracy of 0.5 achieved on the IBM dataset

(2) $F1$ score: the F1 scores for MICE range from 0 to 0.6, with the highest score of 0.6 observed on the IBM dataset

(ii) CPA imputation:

(1) Accuracy: CPA demonstrates an average accuracy of 0.62 across all datasets, with the highest accuracy of 0.72 achieved on the HEC dataset

(2) $F1$ score: the $F1$ scores for CPA range from 0.65 to 0.75, with the highest score of 0.75 observed on the IBM dataset

Overall, the results indicate that both MICE and CPA imputation methods perform reasonably well. CPA generally outperforms MICE in terms of accuracy and $F1$ score, achieving higher values on most datasets. However, it is important to note that the performance varies across datasets, with IBM consistently yielding higher accuracy and $F1$ scores for both imputation methods.

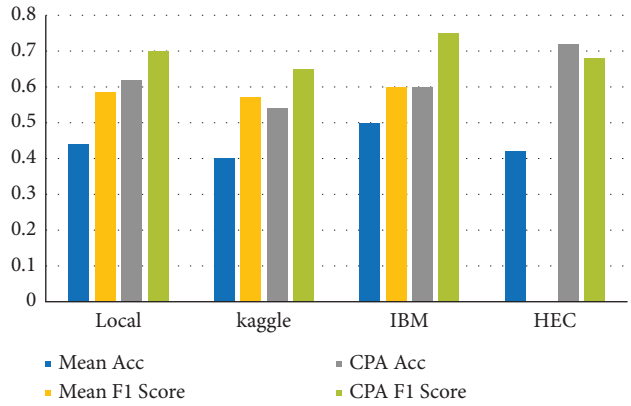

Figure 3: Accuracy and $F1$ score of all datasets using CPA and mean.

Table 8: Multiple imputations and comparison.

		Local	Kaggle	IBM	HEC
MICE	Acc	0.44	0.4	0.5	0.42
	$F1$ score	0.39	0.57	0.6	0
Proposed (CPA)	Acc	0.62	0.54	0.6	0.72
	$F1$ score	0.7	0.65	0.75	0.68

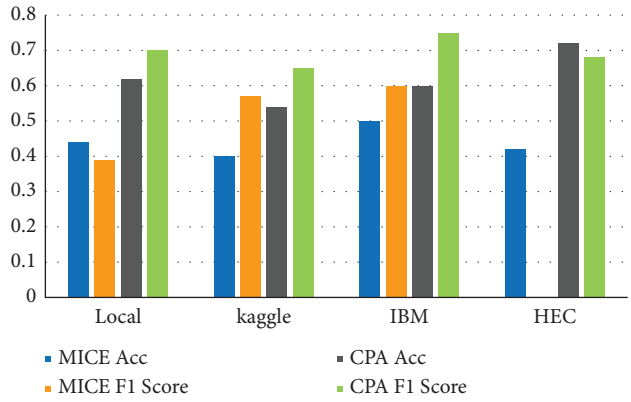

Figure 4: Multiple imputations with comparison.

These results provide insights into the effectiveness of the two imputation techniques on the given datasets, highlighting the importance of choosing the appropriate method based on the specific dataset characteristics and the desired evaluation metrics.

4.3. Machine Learning-Based Imputation. For the third experimental setup, several machine learning approaches were used. Results from those experiments are as shown in the following figures and tables.

Figure 5 displays the predicted accuracy and $f1$ score for the "Age" attribute from both the primary dataset and the IBM dataset. Table 9 presents the results of different classification algorithms, including proposed, SVM, LDA, and DTC, applied to the IBM and local datasets for age prediction. Two evaluation metrics, namely accuracy and $F1$ score, are used to assess the performance of these algorithms.

Comparing the performance of the algorithms on the local dataset, the proposed algorithm demonstrates a high accuracy of 0.92 and an $F1$ score of 0.9474, indicating its ability to predict age effectively. The SVM algorithm achieves an accuracy of 0.88 and an $F1$ score of 0.9081, showing its competence in age prediction. The LDA algorithm also performs reasonably well with an accuracy of 0.87 and an $F1$ score of 0.8978. Lastly, the DTC algorithm exhibits an accuracy of 0.91 and an $F1$ score of 0.9387, suggesting its reliability in age prediction for the local dataset.

Overall, the proposed algorithm performs consistently well across both the IBM and local datasets, demonstrating its robustness and suitability for age prediction. The SVM and DTC algorithms also exhibit strong performance, while the LDA algorithm shows slightly lower accuracy and $F1$ scores but still maintains acceptable predictive capabilities.

4.4. Deep Learning- and Ensemble Learning-Based Imputations. In the final setup, deep learning-based imputations and ensemble learning-based imputations were experimented.

Table 10 shows the results of deep learning-based methods and ensemble learning-based methods for "Gender" attribute in the Kaggle employee-attrition dataset and IBM employee dataset. When considering the $F1$ scores, which provide a measure of the model's ability to balance precision and recall, the proposed CPA algorithm again demonstrated superior performance (visualised in Figure 6). It achieved an $F1$ score of 0.76 on the Kaggle dataset and 0.65 on the IBM dataset, surpassing the other models. MLP had the highest $F1$ score of 0.65 on the IBM dataset, while the other models scored considerably lower or zero.

5. Discussion

In this study, we showed a technique for imputing employee missing data which is based on the probability of the attributes in the dataset. After many experiments, the results indicate that the CPA technique outperforms most of the well-known imputation and prediction algorithms across the field. It is important to consider other factors such as computational efficiency, interpretability, and scalability when selecting the appropriate algorithm for age prediction in specific scenarios. Additionally, further evaluation and experimentation are recommended to validate the generalizability of these algorithms across different datasets and real-world applications. The major difference between the proposed method and existing ones is the emphasis on probabilistic approaches. Unlike traditional methods that might use deterministic imputation techniques, the proposed approach employs probabilistic methods that incorporate uncertainty and variability in the imputation process. This ensures more realistic and reliable estimates of missing data, enabling better identification of patterns and relationships between employee characteristics and attrition.

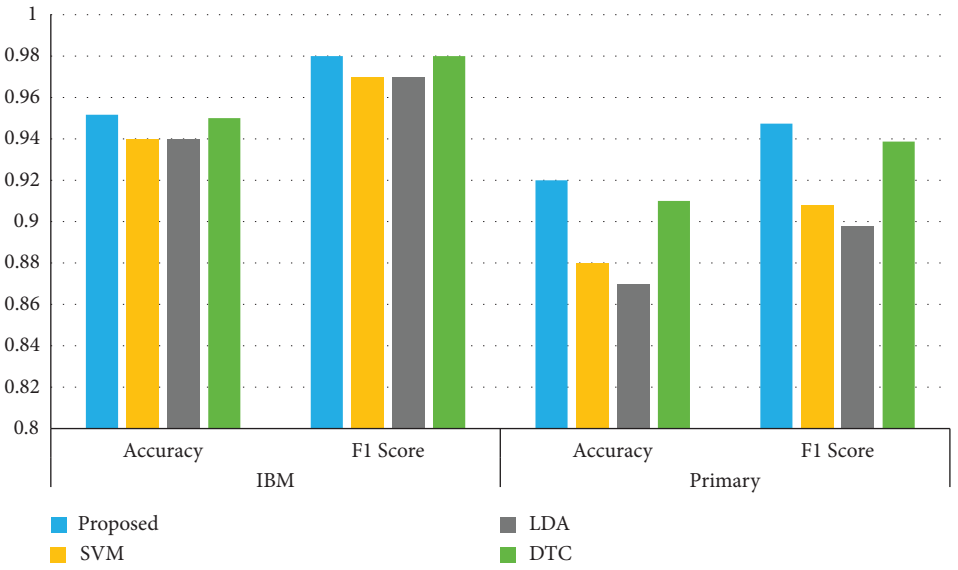

Figure 5: Result analysis of ML algorithms.

Table 9: Result comparison with ML algorithms.

	IBM		Primary	
	Accuracy	*F*1 score	Accuracy	*F*1 score
Proposed (CPA)	0.95	0.98	0.92	0.9474
SVM	0.94	0.97	0.88	0.9081
LDA	0.94	0.97	0.87	0.8978
DTC	0.95	0.98	0.91	0.9387

Table 10: Result comparison with deep learning and ensemble algorithms.

	Kaggle		IBM	
	Accuracy	*F*1 score	Accuracy	*F*1 score
Proposed (CPA)	0.62	0.76	0.54	0.65
ANN	0.56	0.44	0.41	0.07
MLP	0.47	0.0	0.48	0.65
XGB	0.47	0.48	0.46	0.37
Adaboost	0.47	0.0	0.42	0.16

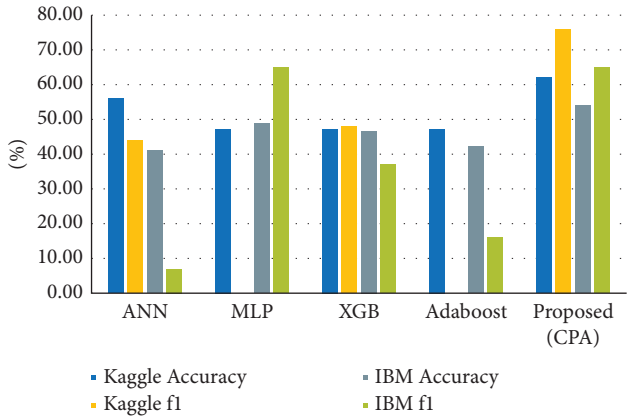

Figure 6: Result analysis of deep learning and ensemble algorithms.

6. Conclusion

Missing data are widespread in real-world datasets, causing a slew of issues in data processing. Several efforts have been made to address this problem for the medical dataset; however, relatively few efforts have been detected for the employee dataset. Data might be omitted in employee datasets for a variety of reasons, one of which is a system failure, and another is data input errors. Employee data analysis is one of the most significant tasks for the economy in order to gain a clear picture of the many components of work in order to cope with future issues. We focused on employee datasets in this study to address missing data prediction. We concentrated on the properties with missing values in the dataset after finishing general preprocessing tasks. The main task is to create a combination from chosen attributes and to determine the probability for each combination node. A subsequent table is made to fill in the missing values based on the analytical findings from the values of each node's probabilities.

For the tested dataset, our study demonstrates the smallest inaccuracy in projected values. Also, when compared to other procedures, this methodology produces better results. Despite the fact that only categorical data may be used in this approach, the generated data can be transformed into numerical utilizing a few well-known techniques. This research may be expanded by lowering the computation cost. The findings of this study can be applied in various industries, including human resources, finance, and management, where employee data analysis plays a crucial role in decision-making processes.

For the tested dataset, our study demonstrates the smallest inaccuracy in projected values. Also, when compared to other procedures, this methodology produces better results. Despite the fact that only categorical data may be used in this approach, the generated data can be transformed into numerical utilizing a few well-known techniques. This

research may be expanded by lowering the computation cost.

Data Availability

(1) The employee-attrition data used to support the findings of this study have been deposited in the Kaggle repository: https://www.kaggle.com/colearninglounge/employee-attrition. (2) The ibm hr analytics attrition dataset used to support the findings of this study has been deposited in the Kaggle repository: https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset. (3) The hair-eye color dataset used to support the findings of this study has been deposited in the Kaggle repository: https://www.kaggle.com/datasets/jasleensondhi/hair-eye-color. (4) The local employee dataset used to support the findings of this study has been deposited in the following one drive link: https://iiucacbd-my.sharepoint.com/:x:/g/personal/nazmul_arefin_faculty_iiuc_ac_bd/EdibpegNiC1Gt8p-cdOZV0cBM_Tyn0V0fPIEBprOiVTKwg?e=q5nqSf.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] G. Liebchen and M. Shepperd, "Data sets and data quality in software engineering," in *Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, Ciudad Real, Spain, September, 2016.

[2] Z. Zhang, "Missing values in big data research: some basic skills," *Annals of Translational Medicine*, vol. 3, no. 21, p. 323, 2015.

[3] G. Ridder and R. A. Moffitt, *Chapter 75 the Econometrics of Data Combination*, Elsevier eBooks, Amsterdam, Netherlands, 2007.

[4] N. Mittag, *Imputations: Benefits, Risks and A Method For Missing Data*, National Institute of Statistical Sciences, Triangle Park, NA, USA, 2013.

[5] J. W. Grzymala-Busse and W. J. Grzymala-Busse, "Handling missing attribute values," *Data Mining and Knowledge Discovery Handbook*, Springer Science & Business Media, Berlin, Germany, pp. 37–57, 2005.

[6] J. Grzymala-Busse, W. Grzymala-Busse, and L. Goodwin, "A comparison of three closest fit approaches to missing attribute values in preterm birth data," *International Journal of Intelligent Systems*, vol. 17, no. 2, pp. 125–134, 2002.

[7] B. Agbo, Y. Qin, and R. Hill, "Best fit missing value imputation (BFMVI) algorithm for incomplete data in the internet of things," in *Proceeding of the 5th International Conference on Internet of Things, Big Data and Security*, January 2020.

[8] P. Keerin, W. Kurutach, and T. Boongoen, "A cluster-directed framework for neighbour based imputation of missing value in microarray data," *International Journal of Data Mining and Bioinformatics*, vol. 15, no. 2, p. 165, 2016.

[9] N. A. Samat and M. N. M. Salleh, "A study of data imputation using fuzzy C-means with particle swarm optimization," in *Advances in Intelligent Systems and Computing*, pp. 91–100, 2016.

[10] A. M. Sefidian and N. Daneshpour, "Missing value imputation using a novel grey based fuzzy c-means, mutual information based feature selection, and regression model," *Expert Systems With Applications*, vol. 115, pp. 68–94, 2019.

[11] B. M. Patil, R. C. Joshi, and D. Toshniwal, "Missing value imputation based on K-mean clustering with weighted distance," in *Communications in Computer and Information Science*, pp. 600–609, 2010.

[12] H. Zhang, P. Xie, and E. Xing, "Missing value imputation based on deep generative models," 2018, https://arxiv.org/abs/1808.01684.

[13] L. Li, J. Zhang, Y. Wang, and B. Ran, "Missing value imputation for traffic-related time series data based on a multi-view learning method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, pp. 2933–2943, 2019.

[14] B. Conroy, L. J. Eshelman, C. Potes, and M. Xu-Wilson, "A dynamic ensemble approach to robust classification in the presence of missing data," *Machine Learning*, vol. 102, no. 3, pp. 443–463, 2015.

[15] T. Thomas and E. Rajabi, "A systematic review of machine learning-based missing value imputation techniques," *Data Technologies and Applications*, vol. 55, no. 4, pp. 558–585, 2021.

[16] G. Madhu, B. L. Bharadwaj, G. Nagachandrika, and K. A. Vardhan, "A novel algorithm for missing data imputation on machine learning," in *Proceedings of the 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, November 2019.

[17] M. Kokla, J. K. Virtanen, M. Kolehmainen, J. Paananen, and K. Hanhineva, "Random forest-based imputation outperforms other methods for imputing LC-MS metabolomics data: a comparative study," *BMC Bioinformatics*, vol. 20, no. 1, p. 492, 2019.

[18] S. Al-Janabi and A. F. Alkaim, "A nifty collaborative analysis to predicting a novel tool (DRFLLS) for missing values estimation," *Soft Computing*, vol. 24, no. 1, pp. 555–569, 2020.

[19] E.-L. Silva-Ramírez, R. Pino-Mejías, M. López-Coello, and M.-D. Cubiles-De-La-Vega, "Missing value imputation on missing completely at random data using multilayer perceptrons," *Neural Networks*, vol. 24, no. 1, pp. 121–129, 2011.

[20] R. Pan, T. Yang, J. Cao, K. Lu, and Z. Zhang, "Missing data imputation by K nearest neighbours based on grey relational structure and mutual information," *Applied Intelligence*, vol. 43, no. 3, pp. 614–632, 2015.

[21] H. Liu, Y. Wang, and W. Chen, "Three-step imputation of missing values in condition monitoring datasets," *IET Generation, Transmission and Distribution*, vol. 14, no. 16, pp. 3288–3300, 2020.

[22] Kaggle, "Employee Attrition," 2023, https://www.kaggle.com/datasets/colearninglounge/employee-attrition.

[23] Kaggle, "IBM HR Analytics Employee Attrition & Performance," 2023, https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset.

[24] Kaggle, "Hair eye color," 2023, https://www.kaggle.com/datasets/jasleensondhi/hair-eye-color.

[25] S. I. Khan and A. S. M. L. Hoque, "SICE: an improved missing data imputation technique," *Journal of Big Data*, vol. 7, no. 1, p. 37, 2020.

[26] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise Reduction in Speech Processing*, Springer Science & Business Media, pp. 1–4, Berlin, Germany, 2009.

[27] S. V. Buuren and K. Groothuis-Oudshoorn, "mice: multivariate imputation by chained equations inr," *Journal of Statistical Software*, vol. 45, no. 3, 2011.

[28] S. Utukuru, R. K. Pisipati, and K. Karlapalem, "Missing data resilient ensemble subspace decision Tree Classifier," in *Proceedings of the 6th Joint International Conference on Data Science; Management of Data (10th ACM IKDD CODS and 28th COMAD)*, Mumbai, India, January, 2023.

[29] S. Moghtadernejad, Y. Jin, and B. T. Adey, "Estimating the values of missing data related to infrastructure condition states using their spatial correlation," *Journal of Infrastructure Systems*, vol. 29, no. 1, 2023.

[30] I. Izonin, N. Kryvinska, R. Tkachenko, and K. Zub, "An approach towards missing data recovery within IOT smart system," *Procedia Computer Science*, vol. 155, pp. 11–18, 2019.

[31] A. Islam, "Ensemble machine learning approach for agricultural crop selection," in *Proceedings of the 2023 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Kolkata, India, January, 2023.

[32] E. Foster and S. Godbole, *Database Systems: A Pragmatic Approach*, Auerbach Publications, Boca Raton, FL, USA, 3rd edition, 2022.

[33] F. Fallucchi, M. Coladangelo, R. Giuliano, and E. William De Luca, "Predicting employee attrition using machine learning techniques," *Computers*, vol. 9, no. 4, p. 86, 2020.

[34] Z. Wang, L. Wang, Y. Tan, and J. Yuan, "Fault detection based on Bayesian network and missing data imputation for Building Energy Systems," *Applied Thermal Engineering*, vol. 182, Article ID 116051, 2021.

[35] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.