Learning Accurate Models on Incomplete Data with Minimal Imputation

Anonymous submission

Abstract

Missing data often exists in real-world datasets, requiring significant time and effort for imputation to learn accurate machine learning (ML) models. In this paper, we demonstrate that imputing all missing values is not always necessary to achieve an accurate ML model. We introduce the concept of minimal data imputation, which ensures accurate ML models trained over the imputed dataset. Implementing minimal imputation guarantees both minimal imputation effort and optimal ML models. We propose algorithms to find exact and approximate minimal imputation for various ML models. Our extensive experiments indicate that our proposed algorithms significantly reduce the time and effort required for data imputation.

1 Introduction

The performance of a machine learning (ML) model heavily depends on the quality of its training data. In real-world data, a major data quality issue is missing data, i.e., incomplete data. There are two prevalent approaches to address missing values in training data. The first approach involves deleting the data points or features with missing data. However, this method can lead to the loss of important information and create bias (Van Buuren 2018).

Another popular approach is imputation in which ML practitioners replace missing values with correct ones (Little and Rubin 2002; Le Morvan et al. 2021; Whang et al. 2023; Farhangfar, Kurgan, and Dy 2008; Andridge and Little 2010; Śmieja et al. 2018; Williams et al. 2005; Pelckmans et al. 2005). However, accurate imputation is usually challenging and expensive as it often requires lengthy collaboration with costly domain experts to find the correct values for missing data items. To reduce the cost of imputation, there has been a significant effort to devise models that predict accurate values for the missing data items (Le Morvan et al. 2021). Nonetheless, it is not clear if these models can deliver effective results and confidently replace domain experts in every domain, particularly in sensitive ones such as medical sciences. It also takes a great deal of time and effort to find, train, and validate proper models for the task at hand. Because their predicted values for missing data are estimations of correct values, using them over a dataset with many missing data items may introduce considerable inaccuracies or biases to the training data, which results in learning an inaccurate or biased model. As accurate imputation models are often quite complex, e.g., DNNs (Mattei and Frellsen 2019), it often takes a significant time to impute a large dataset with missing values using these models. Surveys indicate that ML practitioners spend about 80% of their time preparing data, including imputing missing data (Krishnan et al. 2016; Neutatz et al. 2021).

There have been some efforts to eliminate the burden of data imputation for model training. Researchers have proposed *stochastic optimization* to find a model by optimizing the expected loss function over the probability distributions of missing data items in the training examples (Ganti and Willett 2015). Similarly, in *robust optimization*, researchers minimize the loss function of a model for the imputation that brings the highest training loss while assuming certain distributions of the missing values (Aghasi, Feizollahi, and Ghadimi 2022). However, the distributions of missing data items are not often available. Thus, users may spend significant time and effort to discover or train these distributions, which may require the user to find the causes of missingness in the data and dependencies between the features. Additionally, for a given type of model, users must solve various and possibly challenging optimization problems for many possible (combinations of) distributions of missing values. More importantly, these methods reflect the uncertainty in the training data caused by missing values in the trained model. Hence, they deliver inaccurate models over the dataset with considerably many missing values.

There has been also some work on detecting cases where imputing missing data is not necessary to learn accurate models (Picado et al. 2020; Karlaš et al. 2020; Zhen et al. 2024). These methods check whether an accurate model can be learned without using training examples with missing features. Although these approaches are useful for some datasets and learning tasks, they ignore a majority of learning tasks in which imputing incomplete examples impacts the quality of the learned model.

To address these challenges, we introduce the concept of a **minimal imputation** set for an incomplete training dataset. This set represents the smallest group of missing data items that, once imputed, yields the same model as one trained on a fully and correctly imputed dataset. We propose novel and efficient methods to find minimal imputation for some of the popular ML models. By finding and only imputing the minimal imputation set, users can generally train accurate models over incomplete datasets with significantly less overhead than available imputation methods. The resulting model will generally deliver a more accurate model than methods based on stochastic or robust optimization approaches as it eliminates all relevant uncertainties in the training dataset. Specifically, our contributions are:

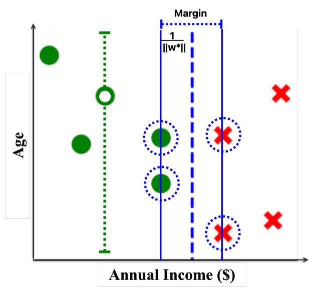
- We formally define the concept of minimal imputation for learning over incomplete data.
- We prove that finding the minimal imputation for support vector machines (SVM) is NP-hard. We propose an efficient method with provable error bounds to approximate minimal imputation for SVM (Section 3).
- We prove that finding the minimal imputation for linear regression is NP-hard. We propose an efficient algorithm with provable error bounds to approximate minimal imputation for linear regression (Section 4).
- We conduct experiments to show cost savings in data cleaning and program execution time compared to imputing all missing data using multiple methods, including mean imputation, a KNN-based imputation algorithm, a deep learning-based imputation method, and a benchmark framework. We also extend the comparison to diverse real-world datasets with inherent missing values, yielding results consistent with randomly corrupted datasets. Our studies show that our algorithms significantly reduce data imputation costs by finding minimal imputations (Section 5).

2 Background

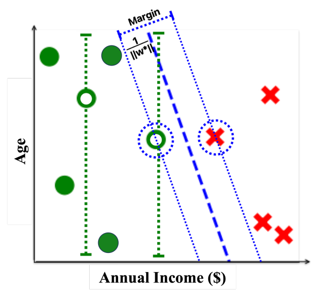
Model Training We model the input training data as a table where each row represents a training example, e.g., Table 1. One column in the table represents the label and others represent features of the examples. Given that the training data has d features, we denote its features as $[\mathbf{z}_1, \ldots, \mathbf{z}_d]$. The values of each feature belong to the *domain of the feature*, e.g., real numbers. To simplify our analysis, we assume that all features share the same domain. Our results extend to other settings. A *training set* with n examples is a pair of a feature matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]^T$ and a corresponding label vector $\mathbf{y} = [y_1, ..., y_n]^T$. We denote each example with d features in $\mathbf{X}$ as a vector $\mathbf{x}_i = [x_{i1}, ..., x_{id}]$, where x_{ij} represents the j^{th} feature in the i^{th} example. Taking a feature matrix $\mathbf{X}$, a label vector $\mathbf{y}$, a target function f, and a loss function L, the goal of training is to find an optimal model $\mathbf{w}^*$ that minimizes the training loss, i.e., $\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathcal{W}} L(f(\mathbf{X}, \mathbf{w}), \mathbf{y})$.

Table 1: A dataset for credit card approval prediction

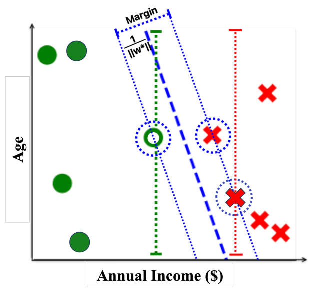
Annual Income($)	Age	Decision
65000	30	1
10000	*null*	-1



(a) Imputing missing values is not necessary at all



(b) Some missing values need imputation while others do not



(c) All missing values need imputation

Figure 1: The necessity of missing data imputation varies

Missing values Any x_{ij} is a missing value if it is unknown (marked by *null*). An *incomplete example* is an example with at least one missing value. Similarly, an *incomplete feature* is a feature that contains at least one missing value. We use *complete feature* and *complete example* to refer to features and examples that are free of missing values. We further denote the set of all missing values in a feature matrix $\mathbf{X}$ as $MV(\mathbf{X})$, the set of incomplete examples as $MVE(\mathbf{X})$, and the set of incomplete features as $MVF(\mathbf{X})$. In this paper, we focus on the case that all missing values are in the feature matrix, and the label vector is complete.

Example 1 *In Table 1, Age is an incomplete feature while Annual Income is a complete one. Similarly, the first training example is complete and the second training example is incomplete.*

Repair A repair is a complete version of an incomplete feature matrix where all missing values are imputed, i.e. replaced with values from their domains. More formally:

Definition 1 *(Repair) $\boldsymbol{X}^r$ is a repair to feature matrix $\boldsymbol{X}$ if 1) $dimension(\boldsymbol{X}^r) = dimension(\boldsymbol{X})$, 2) $\forall x^r_{ij} \in \boldsymbol{X}^r, x^r_{ij} \neq$ null, and 3) $\forall x_{ij} \neq$ null, $x^r_{ij} = x_{ij}$.*

Given the repair $\mathbf{X}^r$ of the feature matrix $\mathbf{X}$, we denote the repair, i.e., imputation, of example $\mathbf{x}_i$ in $\mathbf{X}$ by $\mathbf{x}^r_i$.

Example 2 *In Table 1, replacing the missing data with a value,e.g., 45, produces a repair. However, deleting the Age feature, which eliminates the missing value, is not a repair since it changes $dimension(\boldsymbol{X})$.*

Possible Repairs Since the domains of features often contain numerous or infinite values, an incomplete feature matrix usually has numerous or infinitely many repairs. We denote this set of all repairs of a feature matrix $\mathbf{X}$ as $\mathbf{X}^R$.

Example 3 *Figures 1a, 1b, and 1c display three sets of training examples with missing Age values. The green and red dashed lines represent the range of possible values for the incomplete feature (denoted by empty circles). In Figure 1a, the incomplete examples do not intersect the SVM margins (outlined by the solid blue lines) with any possible repair to the feature matrix ($\boldsymbol{X}^r \in \boldsymbol{X}^R$) and, therefore, are not support vectors. As a result, the model (decision boundary: blue dashed line) remains optimal across all repairs, and a certain model ($\boldsymbol{w}^*$) exists. In this scenario, imputing missing data is unnecessary. However, in Figures 1b and 1c, since at least one incomplete example may intersect the blue rectangle with some repairs of the feature matrix, the optimal model changes from one repair to another, and a certain model does not exist. In this scenario, data imputation is necessary to achieve an accurate model.*

Certain Models Researchers have defined the concept of *certain models* to determine the necessity of imputing missing data for learning (Zhen et al. 2024).

Definition 2 *(Certain Model) A model $\boldsymbol{w}^*$ is a certain model for target function f over training set $(\mathbf{X}, \mathbf{y})$ if:*

$$\forall \boldsymbol{X}^r \in \boldsymbol{X}^R, \boldsymbol{w}^* = \arg\min_{\boldsymbol{w} \in \mathcal{W}} L(f(\boldsymbol{X}^r, \boldsymbol{w}), \boldsymbol{y}) \quad (1)$$

Intuitively, a certain model is optimal, i.e., minimizes the training loss, for all repairs to the incomplete feature matrix. Thus, if a certain model exists, one can learn an accurate model over the dataset without any costly imputation as training over any imputation of the dataset, e.g., using randomly selected values, will deliver the same accurate model. However, given the restrictive definition, certain models do not often exist (Zhen et al. 2024). In this paper, we find the minimal amount of data imputation of an incomplete training set such that the resulting dataset has a certain model.

3 Minimal Imputation for SVM

In SVM, support vectors are the training examples deciding the optimal model which minimizes the loss function $L(f(\mathbf{X}, \mathbf{w}), \mathbf{y}) = \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{i=1}^{n} max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\}$. $(\mathbf{x}_i, y_i)$ is a support vector if $y_i \mathbf{w}^T \mathbf{x}_i \leq 1$. Intuitively, an incomplete example needs to be imputed if it determines the optimal model, i.e., if it is a support vector in some repairs. Conversely, an incomplete example does not need to be imputed if it is not a support vector in any repair. Therefore, we propose a formal definition of the minimal imputation set on an example-wise basis—this is the smallest set of incomplete examples that, when imputed, guarantees an accurate SVM model.

Definition 3 *(Minimal imputation set) A set of incomplete examples $\mathbf{S}$ in the training set $(\mathbf{X}, \mathbf{y})$ is a minimal imputation set for learning SVM with regularization parameter C if we have: 1) a certain model exists upon imputing all missing values $\mathbf{S}$ and 2) there is no set $\mathbf{S}'$ satisfying condition (1) such that $|\mathbf{S}'| < |\mathbf{S}|$ where $|\mathbf{S}|$ denotes the cardinality of $\mathbf{S}$.*

We denoted the minimal imputation set for SVM with regularization parameter C over the training set $(\mathbf{X}, \mathbf{y})$ as $\mathbf{S}_{min}(\mathbf{X}, \mathbf{y}, C)$. We have the following property for the minimal imputation set.

Theorem 1 *(Uniqueness of the minimal imputation set) Given the training set training set $(\mathbf{X}, \mathbf{y})$ and regularization parameter C, $\mathbf{S}_{min}(\mathbf{X}, \mathbf{y}, C)$ is unique.*

3.1 Finding minimal imputation

Let $SV(\mathbf{X^r}, \mathbf{y}, C)$ be the set of support vectors for the optimal SVM model with regularization parameter C over a repair of the training set $(\mathbf{X}, \mathbf{y})$. The following theorem formalizes an important property of the minimal imputation set for SVM discussed at the beginning of this section.

Theorem 2 *Given the training set $(\mathbf{X}, \mathbf{y})$ and regularization parameter C, at least one imputation of every example in $S_{min}(\mathbf{X}, \mathbf{y}, C)$ is a support vector in a repair of $\mathbf{X}$, i.e.,*

$$S_{min}(\mathbf{X}, \mathbf{y}, C) = \{\mathbf{x}_i | \exists \mathbf{X}^r \in \mathbf{X}^R, \mathbf{x}_i^r \in SV(\mathbf{X}, \mathbf{y}, C)\}$$

where $\mathbf{x}_i^r$ is the corresponding repair of $\mathbf{x}_i$ in $\mathbf{X}^r$.

To determine if an incomplete example belongs to the minimal imputation set, one could materialize every repair of the feature matrix and check if the incomplete example is a support vector for any of them. However, this process can be extremely inefficient due to the often large number of repairs. Assume that each missing value x_{ij} is bounded by an interval based on its domain.

Definition 4 *(Edge repair) $\mathbf{X}^e$ is an edge repair to $\mathbf{X}$ if for every missing values x_{ij}, $x_{ij}^e = x_{ij}^{min}$ or x_{ij}^{min}. $\mathbf{X}^E$ denotes the set of all edge repairs of $\mathbf{X}$.*

Theorem 3 shows that we can use only the edge repairs instead of all repairs to check if an incomplete example belongs to the minimal imputation set.

Theorem 3 *Given training set $(\mathbf{X}, \mathbf{y})$ and regularization parameter C, an incomplete example $\mathbf{x}_i$ belongs to the minimal imputation set $S_{min}(\mathbf{X}, \mathbf{y}, C)$ if and only if there is at least one edge repair $\mathbf{X}^e$ of $\mathbf{X}$ such that $\mathbf{x}_i^e \in SV(\mathbf{X}^e, \mathbf{y}, C)$ where $\mathbf{x}_i^e$ is the imputation of $\mathbf{x}_i$.*

Based on Theorem 3, if the number of missing values is small, we can find the minimal imputation set by following these steps: 1) Initialize an empty minimal imputation set, S_{min}. 2) Iterate over each incomplete example $\mathbf{x}_i$. At each iteration, materialize all edge repairs $\mathbf{X}^e \in \mathbf{X}^E$, and check if $\mathbf{x}_i$ is a support vector in any of the edge repairs for any of the edge repairs. If it is, add $\mathbf{x}_i$ to S_{min}, and 3) Finally, return the minimal imputation set S_{min}.

3.2 Approximating Minimal Imputation

If the number of missing values is large, the cost of materializing all edge repairs becomes substantial, as the number of edge repairs grows exponentially with the number of missing values. In this case, we propose an efficient approximation method to find the minimal imputation set.

Instead of exhaustively checking all possible edge repairs to the feature matrix to determine if the target incomplete example $(\mathbf{x}_i)$ is a support vector for any of the edge repairs, we construct an edge repair that approximately maximizes the likelihood of $\mathbf{x}_i$ being a support vector by individually scanning the missing values. More precisely, the process starts with a random edge repair to the feature matrix and then iterates over each missing value. During each iteration, all other repairs are fixed except for one missing value, x_{pq}. We then update x_{pq} to either x_{pq}^{min} or x_{pq}^{max}, whichever minimizes $y_i\mathbf{w}^T\mathbf{x}_i$, thereby making $\mathbf{x}_i$ most likely to be a support vector. By the end of the iteration, we obtain an edge repair, $\mathbf{X}^e$, that approximately maximizes the likelihood of $\mathbf{x}i$ being a support vector. If $\mathbf{x}_i$ is not a support vector in the model trained with $\mathbf{X}^e$, we conclude that $\mathbf{x}i$ is not part of the minimal imputation set S_{min}. Algorithm 1 demonstrates this approach.

Algorithm 1: Approximating minimal imputation for SVM efficiently

```
$S_{min} \leftarrow [\quad]$
$MV(X) \leftarrow$ set of missing values
$MVE(\mathbf{x}) \leftarrow$ set of incomplete examples
$\mathbf{X}^e \leftarrow$ a random edge repair to the feature matrix $\mathbf{X}$
for $\mathbf{x}_i \in MV(\mathbf{x})$ do
    for $x_{pq} \in MV$ do
        $\mathbf{X}^e(x_{pq}^{min}), \mathbf{X}^e(x_{pq}^{max}) \leftarrow$ two edge repairs
        ▷ only replacing $x_{pq}$ in $\mathbf{X}^e$ by its two edge repairs
        $\mathbf{w}_1 \leftarrow SVM(\mathbf{X}^e(x_{pq}^{min}), \mathbf{y})$
        $\mathbf{w}_2 \leftarrow SVM(\mathbf{X}^e(x_{pq}^{max}), \mathbf{y})$
        if $y_i\mathbf{w}_1^T \leq y_i\mathbf{w}_2^T$ then
            $\mathbf{X}^e \leftarrow \mathbf{X}^e(x_{pq}^{min})$
        end if
    end for
    $\mathbf{w} \leftarrow SVM(\mathbf{X}^e, \mathbf{y})$
    if $y_i\mathbf{w}^T\mathbf{x}_i \leq 1$ then
        $S_{min} \leftarrow S_{min}.add(\mathbf{x}_i)$
    end if
end for
return $S_{min}$   ▷ The minimal example set for imputation
```

Since we modify only one missing value at each iteration, the datasets used to train adjacent models ($\mathbf{w}_1$ and $\mathbf{w}_2$) differ by just a single value in the feature matrix. Therefore, instead of fully retraining a new model for each iteration, we can leverage incremental and decremental learning to significantly reduce computational costs. Specifically, given an SVM model $\mathbf{w}$ and its corresponding feature matrix $\mathbf{X}$, if we change only one value x_{ij} to create a new feature matrix $\mathbf{X}'$, we can obtain the updated SVM model $\mathbf{w}'$ for $\mathbf{X}'$ through incremental and decremental learning with a time complexity that is one order of magnitude lower than that required for fully retraining the model (Laskov et al. 2006; Cauwenberghs and Poggio 2000).

Next. we analyze the accuracy of Algorithm 1. First, Algorithm 1 does not return any incomplete example that does not belong to the minimal imputation set of the training data, i.e., no false positives.

Theorem 4 *Every example returned by Algorithm 1 over training set $(\mathbf{X}, \mathbf{y})$ and regularization parameter C belongs to $S_{min}(\mathbf{X}, \mathbf{y}, C)$.*

However, Algorithm 1 may miss some examples that are in the minimal imputation set, i.e., false negative. The following theorem bounds the probability of such an event.

Theorem 5 *Let $g(x_{ij})$ denote the probability density function of missing value x_{ij} in incomplete training set X. If missing values in X are independent, the probability that an incomplete example $\mathbf{x}_i$ that belongs to the minimal imputation set of X but not returned by Algorithm 1 is:*

$$p(\mathbf{x}_i) = \frac{\int \cdots \int_{min(x_{ij}^{visited})}^{max(x_{ij}^{visited})} \prod_{x_{ij} \in MV} g(x_{ij})\, dx_{ij}}{\int \cdots \int_{x_{ij} \in MV} \prod_{x_{ij} \in MV} g(x_{ij})}$$

where $x_{ij}^{visited}$ is a set containing the value(s) we have used for x_{ij} in Algorithm 1. $x_{ij}^{visited}$ can either be $\{x_{ij}^{min}\}$, $\{x_{ij}^{max}\}$, or $\{x_{ij}^{min}, x_{ij}^{max}\}$. $min(x_{ij}^{visited})$ and $max(x_{ij}^{visited})$ represent the minimum and maximum values in the set, respectively.

Theorem 5 indicates that, generally, the more edge repairs we cover in Algorithm 1, the higher the likelihood that S_{min} will not miss any incomplete examples.

4 Minimal Imputation for Linear Regression

We define the minimal imputation for linear regression as follows.

Definition 5 *Given the training set $(\mathbf{X}, \mathbf{y})$, a set of incomplete features in $\mathbf{X}$, denoted as $\mathbf{S}_{min}(\mathbf{X}, \mathbf{y})$, is a minimal imputation set for $(\mathbf{X}, \mathbf{y})$ if we have: 1) a certain model exists upon imputing all missing values in the $\mathbf{S}_{min}(\mathbf{X}, \mathbf{y})$, and 2) there is no set $\mathbf{S}$ satisfying condition (1) and $|\mathbf{S}| < |\mathbf{S}_{min}|$.*

Definition's Intuition A minimal set for imputations refers to the smallest set of incomplete features that are necessary to impute to ensure an accurate linear regression model.

In linear regression, the optimal linear regression model $\mathbf{w}^*$ consists of the set of linear coefficients for feature vectors. A feature $\mathbf{z}_i$ is considered relevant if the corresponding linear coefficient in the optimal model w_i^* is not zero, and it is irrelevant if w_i^* equals zero. Intuitively, an incomplete feature needs to be imputed if it is relevant (i.e., it plays a role in the optimal model) and does not need to be imputed if it is irrelevant. However, traditional statistical tools, such as the chi-square test, require complete distributions for each feature to assess correlations, which is challenging in the presence of missing values. Therefore, identifying the minimal set of incomplete features for imputation involves determining which features are irrelevant, regardless of how other incomplete features are handled. Note that the minimal imputation set for linear regression is not necessarily unique.

Theorem 6 *There is a training set with multiple minimal imputation sets for linear regression. Also, if all features in all repairs of the training set $(\mathbf{X}, \mathbf{y})$ are linearly independent, the minimal imputation set for linear regression over $(\mathbf{X}, \mathbf{y})$ is unique.*

4.1 Finding Minimal Imputation

A baseline algorithm for finding the minimal imputation set is: (1) learn linear regression models with each possible repair individually, and (2) identify the set $\mathbf{S}_{min}$ with the smallest size such that all repairs to features not in $\mathbf{S}_{min}$ share at least one mutual optimal model. However, the set of possible repairs is often large, making the process of learning models from all repairs potentially very slow. In our search for a more efficient method, we note that finding the minimal imputation set for linear regression is intractable, as established in Theorem 7

Theorem 7 *Finding the minimal imputation set for linear regression over a training set is NP-hard.*

Therefore, we aim to develop efficient algorithms for finding the minimal imputation set without materializing all repairs.

4.2 Approximating Minimal Imputation Efficiently

We first propose an equivalent problem in Theorem 8, based on a variant of the well-known sparse linear regression problem (Bruckstein, Donoho, and Elad 2009).

Theorem 8 *Finding the minimal imputation set for linear regression over the training set* $(\mathbf{X}, \mathbf{y})$ *is equivalent to:*

$$\min_{\mathbf{w}\in\mathcal{W}} T_{MVF(\mathbf{X})}(\mathbf{w})$$
$$\textit{subject to} \quad \mathbf{w} = \arg\min ||\mathbf{X}^r\mathbf{w} - \mathbf{y}||_2^2, \forall \mathbf{X}^r \in \mathbf{X}^R$$

where $T_{MVF(\mathbf{X})}(\mathbf{w})$ *is the number of non-zero linear coefficient in* $\mathbf{w}$ *whose corresponding feature is complete, i.e.,* $T_{MVF(\mathbf{X})}(\mathbf{w}) = |\{\mathbf{z}_i \in MVF(\mathbf{X}) | w_i! = 0\}|$

The key distinction between our problem and sparse linear regression lies in their objectives: sparse linear regression seeks to minimize the number of non-zero coefficients across all features, whereas our problem focuses on minimizing the number of non-zero coefficients only among incomplete features.

Orthogonal Matching Pursuit (OMP) provides an efficient approximation for solving the sparse linear regression problem (Wang, Kwon, and Shim 2012). Essentially, this greedy algorithm begins with an empty solution set and initializes the regression residual to the label vector. In each iteration, the algorithm selects the feature most relevant to the current residual (i.e., having the largest dot product), adds it to the solution set, retain a linear regression model, and updates the residual accordingly. The program stops when the regression residue is sufficiently small. Therefore, OMP will return a subset of features (the solution set) that are sufficient to achieve an optimal linear regression model.

In this paper, we propose a variant of OMP, as outlined in Algorithm 2, to find minimal imputation for linear regression. Our algorithm has two major differences compared to the conventional OMP. Firstly, we include all complete features in the regression at the initialization, ensuring that we minimize the number of non-zero coefficients only among incomplete features. Secondly, we define our stopping condition by the maximum relevance (cosine similarity) between the feature and the label being smaller than or equal to a user-defined threshold, instead of relying on a near-zero regression residue. This approach enables our algorithm to work with general datasets without requiring the assumption of an underdetermined linear system, which is typically necessary in conventional OMP.

Algorithm 2: Approximating minimal imputation for linear regression efficiently

```
$S_{min} \leftarrow [\ \ ]$
$MVF(\mathbf{z}) \leftarrow$ set of incomplete features
$Complete(\mathbf{z}) \leftarrow$ set of complete features
$\mathbf{r} \leftarrow LR(Complete(\mathbf{z}), \mathbf{y})$
  ▷ The residue vector from performing linear regression between complete features and label
$\epsilon \leftarrow$ a user-defined threshold for stopping condition
$MaxCosSim \leftarrow \max_{\mathbf{z}\in MVF(\mathbf{z})} |cos(\mathbf{z}, \mathbf{r})|$
  ▷ The maximum absolute value of the cosine similarity between the feature and the regression residual
while $MaxCosSim \leq \epsilon$ do
    $S_{min} \leftarrow S_{min}.add(\arg\max_{\mathbf{z}\in MVF(\mathbf{z})} |cos(\mathbf{z}, \mathbf{r})|)$
    $\mathbf{r} \leftarrow LR(Complete(\mathbf{z}) \cup S_{min}, \mathbf{y})$
    $MaxCosSim \leftarrow \max_{\mathbf{z}\in MVF(\mathbf{z})} |cos(\mathbf{z}, \mathbf{r})|$
end while
return $S_{min}$      ▷ The minimal feature set for imputation
```

The time complexity of the algorithm is $\mathcal{O}(T_{train} \cdot |MVF(\mathbf{z})|)$, making it significantly more efficient than the baseline algorithm of training models over all repairs individually, which has a time complexity of $\mathcal{O}(T_{train} \cdot |\mathbf{X}^R|)$. Additionally, the approximation of minimal imputation does not depend on the probability distribution of the missing values. However, if the probability distributions of the missing values are known, the following approximation rate can be achieved under certain conditions, as demonstrated in Theorem 9. This approximation rate tells the probability that our approximation is correct, meaning that the incomplete features identified by Algorithm 2 are part of at least one minimal imputation set (recall that the minimal imputation set is not necessarily unique for linear regression in Theorem 6).

Theorem 9 *The first* k *incomplete features added to* S_{min} *in Algorithm 2 for input training set* $(\mathbf{X}, \mathbf{y})$ *belong to the minimal imputation set of* $(\mathbf{X}, \mathbf{y})$ *with a probability of at least* $1 - 1/n$*, provided the following conditions are met: 1)* $\mu < 1/(2k-1)$*, 2) the missing values in the dataset follow independent zero-mean normal distributions, i.e.,* $\forall x_{ij} = null$*,* $x_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$*, and 3) all linear coefficients (*$w_i, \mathbf{z}_i \in MVF(\mathbf{X})$*) for incomplete features satisfy:*

$$|w_i| \geq \frac{2\sum_{x_{ij}=null} \sigma_{ij}\sqrt{n + 2\sqrt{nlogn}}}{1 - (2k-1)\mu}$$

where μ *is the mutual incoherence defined by*

$$\mu = \max_{i\neq j} |\mathbf{z_i}^T \mathbf{z_j}|$$

Given the probability distribution of missing values, the smaller the mutual incoherence, the more incomplete features are guaranteed to be correctly included in S_{min}.

5 Experimental Evaluation

We conducted experiments on a diverse set of real-world datasets, comparing our algorithms with one natural baseline, a KNN imputation method, a deep learning-based imputation method, and the benchmark method ActiveClean. Our results demonstrate significant reductions in data cleaning costs without compromising ML model performance while maintaining competitive program execution times.

5.1 Experimental Setup

Hardware and Platform The experiments were performed on an x86_64 machine equipped with 30 Intel(R) Xeon(R) CPU E5-2670 v3 cores at 2.30GHz, running in a VMware virtualized environment with two NUMA nodes and a total of 30MB L3 cache.

Real-world Datasets with Randomly Generated Missing Values We conducted SVM experiments on three real-world datasets, originally complete, but corrupted with randomly injected missing values at missing factors (MF) of 1%, 5%, 10%, and 20%. MF represents the ratio of incomplete examples to the total examples. For each MF, we averaged results over three random corruptions where certain models exist to reduce performance variability.

Malware Dataset This dataset differentiates between malware and benign software by analyzing JAR files (Ricardo P Pinheiro et al. 2019). It contains 6825 features and 1996 training examples.

Gisette Dataset The dataset distinguishes between the handwritten digits 4 and 9 (Isabelle Guyon et al. 2003). It consists of 13500 training examples and 5000 features.

TUANDROMD Dataset The dataset detects Android malware versus benign applications (Borah 2020). It includes 4464 training examples with 241 distinct features.

Real-world Datasets with Original Missing Values We also experimented with eight real-world datasets that originally contained missing values, selected from diverse domains with various *missing factors* (Section 5.1). Table 2 summarizes these datasets. For preprocessing, we dropped examples with missing labels and used *sklearn's* OneHotEncoder for categorical attributes.

Water Potability This dataset includes data on substances in freshwater, predicting whether water is potable or not (Kadiwal 2021).

Breast Cancer This dataset contains characteristics of potential cancerous tissue, predicting whether it is benign or malignant (Wolberg 1992).

Online Education The dataset contains survey responses on online education, predicting whether students prefer cellphones or laptops for the courses (Vanschoren et al. 2013).

Algorithms for Comparison In our experiments, we include two natural imputation baselines, a deep learning-based imputation algorithm, and a benchmark algorithm for comparison.

Active Clean(AC): ActiveClean (Krishnan, Sanjay et al. 2018; Krishnan et al. 2016) aims at minimizing the number of repaired examples to achieve an accurate model.

KNN-Imputer(KI): This method predicts the values of missing items based on observed examples using a KNN classifier (Mattei and Frellsen 2019).

Deep-learning based Imputation (DI): We utilize MIWAE (Mattei and Frellsen 2019) that uses deep latent variable models to predict the values of missing data items based on the value of observed examples.

Mean Imputation(MI): Each missing feature value is imputed with the mean value of that feature (Mattei and Frellsen 2019).

Metrics We evaluate all algorithms on a held-out test set with complete examples. We use accuracy as a metric for classification tasks. Accuracy reflects the percentage of total correct class predictions therefore *higher accuracy is preferred*. For regression tasks, we use mean squared error (MSE). MSE measures the average squared difference between actual and predicted values. Therefore, a *lower MSE is preferred*. We also study the algorithms' data cleaning efforts in terms of program execution time and the number of examples cleaned.

5.2 Results on Real-world Datasets with Random Corruption

Table 3 presents the performance comparison between our minimal imputation method and other approaches for SVM across datasets with randomly generated missing values. The results indicate that finding and imputing only the minimal imputation set allows us to clean a much smaller subset of incomplete examples. Compared to the benchmark method, ActiveClean, which also aims to minimize data cleaning operations, our method imputes significantly fewer incomplete examples in most datasets and missing factors. In terms of program running time, finding and imputing the minimal imputation set generally takes longer than other baseline methods, except in certain cases with DI, where DI can take several hours without returning a result (denoted as 'N/A' in the table). However, it's important to reiterate that the majority of data imputation costs typically arise from the time and labor of domain experts in developing an accurate imputation method, rather than from the imputation program's running time. Overall, the additional time required to find and impute the minimal imputation set is often justified by the significant reduction in the number of incomplete examples that need to be imputed. In most datasets (Gisette and Tuandromd), the SVM model accuracy with the minimal imputation method is comparable to that of other baseline imputation methods across all missing factors. This suggests that the algorithm effectively approximates the minimal imputation set, with omitting imputation for examples outside this set having minimal impact on the ML model's performance.

Table 2: Details of Real World Dataset containing missing values

Data Set	Task	Features	Training Examples	Missing Factor
Breast Cancer	Classification	10	559	1.97%
Water-Potability	Classification	9	2620	39.00%
Online Education	Classification	36	7026	35.48%
Air Quality	Regression	12	7344	90.80%
Communities	Regression	1954	1595	93.67%

Table 3: Performance Comparison on Real-World Datasets with Randomly Generated Missing Values

Data Set	MF	**Examples Imputed**			Time (Sec)					Accuracy (%)				
		MI/KI/DI	MinI	AC	MinI	DI	KI	MI	AC	MinI	DI	KI	MI	AC
Gisette	1%	60	**2**	30	307.00	N/A	7.41	0.84	9.0	97.50	N/A	97.20	97.30	**97.94**
	5%	300	**6**	249	1530.62	N/A	29.76	1.06	17.92	**97.20**	N/A	97.16	97.16	97.16
	10%	600	**10**	394	2815.36	N/A	52.61	3.12	57.30	97.60	N/A	97.20	96.30	**99.93**
	20%	1200	**45**	582	14387.77	N/A	116.86	1.09	113.24	97.00	N/A	97.20	97.20	**99.93**
Malware	1%	15	**3**	14	21.12	N/A	1.93	1.46	3.96	70.92	N/A	**80.70**	**80.70**	78.50
	5%	79	**26**	40	111.83	N/A	4.32	1.32	1.40	73.18	N/A	**80.70**	**80.70**	79.83
	10%	159	**64**	66	584.94	N/A	66.86	2.95	60.23	70.93	N/A	**80.70**	**80.70**	**80.70**
	20%	319	125	**68**	1150.30	N/A	129.72	2.96	120.22	52.13	N/A	**80.70**	**80.70**	68.70
Tuandromd	1%	35	34	**30**	0.87	1118.20	0.06	0.02	0.30	**98.65**	**98.65**	**98.65**	**98.65**	**98.65**
	5%	178	**4**	84	4.57	1118.98	0.16	0.03	3.43	98.78	**98.88**	98.65	98.65	98.78
	10%	357	**9**	110	18.41	1122.90	0.62	0.06	7.33	**98.76**	**98.76**	**98.76**	**98.76**	98.60
	20%	714	**14**	267	36.91	1128.98	1.39	0.07	8.19	**98.76**	**98.76**	**98.76**	**98.76**	**98.76**

Table 4: SVM: Performance Comparison on Real-World Datasets with Original Missing Values

Data Set	**Examples Imputed**			Time (Sec)						Accuracy(%)					
	MI/KI/DI	MiniI	AC	MinIM	MinIK	DI	KI	MI	AC	MiniIM	MinIK	DI	KI	MI	AC
Breast Cancer	11	**3**	10	0.004	0.01	15.75	0.002	0.001	0.001	**66.66**	**66.66**	**66.66**	**66.66**	**66.66**	50.55
Water Potability	1022	787	**30**	26.04	27.12	65.14	0.38	0.04	0.40	**60.53**	39.70	**60.53**	39.70	**60.53**	49.15
Online Education	2427	2112	**20**	215.74	218.82	255.87	2.65	0.14	3.30	65.14	**65.23**	**65.23**	**65.23**	**65.23**	52.02

Table 5: Linear Regression: Performance Comparison on Real-World Datasets with Original Missing Values

Data Set	**Examples Imputed**		Time (Sec)					Mean Squared Error (MSE)				
	MI/KI/DI	MiniI	MinIM	MinIK	DI	KI	MI	MiniIM	MinIK	DI	KI	MI
Air Quality	4	2	0.003	0.779	60.91	1.763	0.003	65.47	59.53	63.08	64.17	**72.88**
Communities	25	20	0.12	10.00	2330	9.691	0.141	0.0512	0.4093	**0.0005**	0.5497	0.004

5.3 Results on Real-world Datasets with Inherent Missingness

Tables 4 and 5 present a performance comparison between our minimal imputation method and other approaches for SVM and linear regression, respectively, across datasets with original missing values. Consistent with results from datasets with randomly generated missing values, our method effectively cleans only a small subset of incomplete examples by identifying and imputing the minimal imputation set. In the SVM experiments, while the minimal imputation method results in fewer imputed examples compared to ActiveClean in the Breast Cancer dataset, ActiveClean performs better in the other two datasets. Despite this, the SVM classification accuracy with our minimal imputation method remains consistently close to that of other baseline imputation methods, whereas ActiveClean exhibits significantly lower accuracy across all datasets. We do not include ActiveClean in the linear regression comparison because its implementation does not support linear regression. Our minimal imputation method achieves regression MSE values that are comparable to those of the baseline methods.

6 Related Work

Researchers have proposed methods to reduce imputation costs (Krishnan et al. 2016; Karlaš et al. 2020). ActiveClean learns models using stochastic gradient descent and greedily chooses examples for imputation that may reduce gradient the most (Krishnan et al. 2016). As opposed to our methods, it does not provide any guarantees of minimal imputation. Due to the inherent properties of stochastic gradient descent, it is challenging to provide such a guarantee. CPClean follows a similar greedy idea but it is limited to learning k nearest neighbor models over incomplete data and does not support the types of models we address. It also does not provide any guarantees of minimality for its imputations.

References

Aghasi, A.; Feizollahi, M.; and Ghadimi, S. 2022. RIGID: Robust Linear Regression with Missing Data. *arXiv preprint arXiv:2205.13635*.

Andridge, R. R.; and Little, R. J. 2010. A review of hot deck imputation for survey non-response. *International statistical review*, 78(1): 40–64.

Borah, P. e. a. 2020. Malware Dataset Generation and Evaluation. In *2020 IEEE 4th Conference on Information and Communication Technology (CICT)*, 1–6. IEEE.

Bruckstein, A. M.; Donoho, D. L.; and Elad, M. 2009. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1): 34–81.

Cauwenberghs, G.; and Poggio, T. 2000. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 13.

Farhangfar, A.; Kurgan, L.; and Dy, J. 2008. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41(12): 3692–3705.

Ganti, R.; and Willett, R. M. 2015. Sparse Linear regression with missing data. *arXiv preprint arXiv:1503.08348*.

Isabelle Guyon et al. 2003. Gisette.

Kadiwal, A. 2021. Water Potability. Kaggle.

Karlaš, B.; Li, P.; Wu, R.; Gürel, N. M.; Chu, X.; Wu, W.; and Zhang, C. 2020. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. *arXiv preprint arXiv:2005.05117*.

Krishnan, S.; Wang, J.; Wu, E.; Franklin, M. J.; and Goldberg, K. 2016. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 9(12): 948–959.

Krishnan, Sanjay et al. 2018. Cleaning for Data Science.

Laskov, P.; Gehl, C.; Krüger, S.; Müller, K.-R.; Bennett, K. P.; and Parrado-Hernández, E. 2006. Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research*, 7(9).

Le Morvan, M.; Josse, J.; Scornet, E.; and Varoquaux, G. 2021. What's a good imputation to predict with missing values? In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 11530–11540. Curran Associates, Inc.

Little, R.; and Rubin, D. 2002. *Statistical analysis with missing data*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley. ISBN 9780471183860.

Mattei, P.-A.; and Frellsen, J. 2019. MIWAE: Deep Generative Modelling and Imputation of Incomplete Data Sets. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 4413–4423. PMLR.

Neutatz, F.; Chen, B.; Abedjan, Z.; and Wu, E. 2021. From Cleaning before ML to Cleaning for ML. *IEEE Data Eng. Bull.*, 44(1): 24–41.

Pelckmans, K.; De Brabanter, J.; Suykens, J. A.; and De Moor, B. 2005. Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5-6): 684–692.

Picado, J.; Davis, J.; Termehchy, A.; and Lee, G. Y. 2020. Learning over dirty data without cleaning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 1301–1316.

Ricardo P Pinheiro et al. 2019. REJAFADA.

Śmieja, M.; Struski, Ł.; Tabor, J.; Zieliński, B.; and Spurek, P. 2018. Processing of missing data by neural networks. *Advances in neural information processing systems*, 31.

Van Buuren, S. 2018. *Flexible imputation of missing data*. CRC press.

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2): 49–60.

Wang, J.; Kwon, S.; and Shim, B. 2012. Generalized orthogonal matching pursuit. *IEEE Transactions on signal processing*, 60(12): 6202–6216.

Whang, S. E.; Roh, Y.; Song, H.; and Lee, J.-G. 2023. Data collection and quality challenges in deep learning: A data-centric ai perspective. *The VLDB Journal*, 32(4): 791–813.

Williams, D.; Liao, X.; Xue, Y.; and Carin, L. 2005. Incomplete-data classification using logistic regression. In *Proceedings of the 22nd International Conference on Machine learning*, 972–979.

Wolberg, W. 1992. Breast Cancer Wisconsin (Original). UCI Machine Learning Repository.

Zhen, C.; Aryal, N.; Termehchy, A.; and Chabada, A. S. 2024. Certain and Approximately Certain Models for Statistical Learning. *arXiv preprint arXiv:2402.17926*.

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) **yes**
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) **yes**
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper (yes/no) **yes**

Does this paper make theoretical contributions? (yes/no) yes

If yes, please complete the list below.

- All assumptions and restrictions are stated clearly and formally. (yes/partial/no) **yes**
- All novel claims are stated formally (e.g., in theorem statements). (yes/partial/no) **yes**
- Proofs of all novel claims are included. (yes/partial/no) **yes**
- Proof sketches or intuitions are given for complex and/or novel results. (yes/partial/no) **yes**
- Appropriate citations to theoretical tools used are given. (yes/partial/no) **yes**
- All theoretical claims are demonstrated empirically to hold. (yes/partial/no/NA) **yes**
- All experimental code used to eliminate or disprove claims is included. (yes/no/NA) **yes**

Does this paper rely on one or more datasets? (yes/no) yes

If yes, please complete the list below.

- A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) **yes**
- All novel datasets introduced in this paper are included in a data appendix. (yes/partial/no/NA) **NA**
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/-partial/no/NA) **NA**
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes/no/NA) **yes**
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes/partial/no/NA) **yes**
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (yes/partial/no/NA) **NA**

Does this paper include computational experiments? (yes/no) yes

If yes, please complete the list below.

- Any code required for pre-processing data is included in the appendix. (yes/partial/no) **yes**
- All source code required for conducting and analyzing the experiments is included in a code appendix. (yes/partial/no) **yes**
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no) **yes**
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) **yes**
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes/partial/no/NA) **yes**
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes/partial/no) **yes**
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes/partial/no) **yes**
- This paper states the number of algorithm runs used to compute each reported result. (yes/no) **yes**
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes/no) **no**
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes/partial/no) **yes**
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes/partial/no/NA) **NA**
- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes/partial/no/NA) **NA**