When Can We Ignore Missing Data in Model Training?

Cheng Zhen
Oregon State University
zhenc@oregonstate.edu

Amandeep Singh Chabada
Oregon State University
chabadaa@oregonstate.edu

Arash Termehchy
Oregon State University
termehca@oregonstate.edu

ABSTRACT

Imputing missing data is typically expensive, and as a result, people seek to avoid it when possible. To address this issue, we introduce a method that determines when data cleaning is unnecessary for machine learning (ML). If a model can minimize the loss function regardless of the missing data's actual values, then data cleaning is not required. We offer efficient algorithms for checking this condition in multiple ML problems, and by analyzing the algorithms, we show that data cleaning is unnecessary when dealing with irrelevant and redundant data. Our preliminary experiments demonstrate that our algorithms can significantly reduce cleaning costs compared to a benchmark method, without incurring much computational overhead in many cases.

CCS CONCEPTS

• **Information systems** → **Data cleaning**.

KEYWORDS

data cleaning, machine learning, irrelevant and redundant data

ACM Reference Format:
Cheng Zhen, Amandeep Singh Chabada, and Arash Termehchy. 2023. When Can We Ignore Missing Data in Model Training?. In *Data Management for End-to-End Machine Learning (DEEM '23), June 18, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3595360.3595854

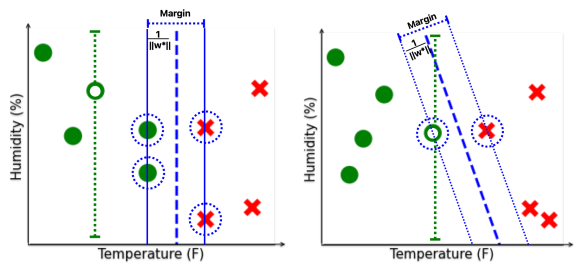
1 INTRODUCTION

The performance of machine learning (ML) relies on the quality of training data. Raw data often has its part missing, bringing a high cost of data cleaning to the ML pipeline: data engineers spend more than 80% of their time cleaning and processing data for ML[1].

The easiest way to handle missing values is by dropping the rows or columns with missing values. Nonetheless, this approach may lose out on useful features or training examples [6]. Another method is to replace missing data with some value, i.e., data imputation. However, data imputation usually requires a significant amount of manual effort by data scientists or domain experts. There are some techniques to reduce human effort in data cleaning for model training [3]. These methods, however, still require experts to spend a significant amount of time and effort to clean subsets of data.

It is sometimes possible to learn an accurate model without cleaning the training data. For instance, if the tuples with erroneous values do not significantly influence the target model, the user can learn an accurate model over raw data without any data cleaning effort. If users can efficiently check these cases, they can learn accurate models without any data preparation effort. This approach has been first successfully applied for learning relational models [5]. Researchers have later used this approach for learning K-nearest neighbor classifiers [2]. However, it remains unclear if these methods can generalize to other ML algorithms.

To overcome this limitation, our goal is to find a generalizable method to check efficiently whether one can learn an accurate model over dirty data. In this paper, we propose an efficient method that checks the necessity of data cleaning over datasets with missing values for a couple of popular ML models. Our method efficiently checks whether there is a model that minimizes training loss irrespective of missing data values



(a) Data cleaning is not needed **(b) Data cleaning is needed**

Figure 1: Data cleaning may not always necessary

Example 1.1. *Our aim is to create a linear hard-margin SVM classifier for rain prediction using temperature and humidity values from different cities. Figures 1a) and 1b) display two sets of data with a missing humidity value, possibly due to a malfunctioning sensor. The green line represents the range of possible values for the missing data, while support vectors are circled in blue. In Figure 1a), the missing value is not a support vector, and the model (indicated by the blue dashed line) can maximize the margin without requiring imputation. However, in Figure 1b), accurate results require data cleaning to obtain the actual value of the missing data for the optimal model.*

In summary, this paper makes the following contributions:

- We formally define the condition where data cleaning is not needed for model training (Section 3).
- We propose efficient algorithms for checking the existence accurate models over datasets with missing values and learning such models for linear regression and SVM problems (Sections 4 and 5).

- We experimentally demonstrate that our algorithm efficiently checks the existence of and learns accurate models over datasets with missing values (Section 8).

The proofs and pseudo-code for algorithms are available in [7].

2 BACKGROUND

This paper presents examples related to supervised learning including linear regression and SVM. The ideas can also be applied to unsupervised learning models as long as they can be formulated as optimization problems.

2.1 Supervised Learning

Training set. Given d features and n training examples, a training set consists of feature input $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_d] = [\mathbf{e}_1, ..., \mathbf{e}_n]^T$ and label output $\mathbf{y} = [y_1, ..., y_n]^T$. $\mathbf{x}_i$, $\mathbf{e}_i$, and y_i are the i^{th} feature vector, the i^{th} training example, and the i^{th} label, respectively. X_{ij} denotes the j^{th} feature input in the i^{th} training example.

Model training. Given a target function f mapping from $\mathbf{X}$ to $\mathbf{y}$, the model training process finds the optimal model $\mathbf{w}^*$ that minimizes training loss $L(f(\mathbf{X}, \mathbf{w}), \mathbf{y})$, i.e., $\mathbf{w}^* = \underset{\mathbf{w} \in \mathcal{W}}{\arg\min}\, L(f(\mathbf{X}, \mathbf{w}), \mathbf{y})$.

2.2 Missing Values and Repairs

Missing values. Any X_{ij} or y_i is a missing value if it is unknown (marked by *null*). I and J are sets of rows and columns with missing values, respectively. $MVX = \{(i,j) | X_{ij} \text{ is missing}\}$ and $MVy = \{i | y_i \text{ is missing}\}$ are index sets of missing values for feature inputs and label outputs, respectively.

Repairs. A repair is a missing-value-free version of the raw data where all missing data are replaced with values.

Definition 1. *For a feature input $\boldsymbol{X}$ having missing values, $\boldsymbol{X}^r$ is a repair to $\boldsymbol{X}$ if 1) $dim(\boldsymbol{X}^r) = dim(\boldsymbol{X})$, 2) $\boldsymbol{X}^r$ does not have missing values, and 3) $\forall (i,j) \notin MVX$, $\boldsymbol{X}^r_{ij} = \boldsymbol{X}_{ij}$.*

Similarly, $\mathbf{y}^r$ is the repair to $\mathbf{y}$ if the label has missing values. An infinite number of repairs may exist. Therefore, we further denote $\mathbf{X}^R$ and $\mathbf{y}^R$ as the union set of all possible repairs to $\mathbf{X}^r$ and $\mathbf{y}^r$, respectively.

Example 2.1. *Table 1 is a dummy training set comprising two features, temperature and humidity, and rain/no rain label. Notably, the humidity value for the New York entry is missing. Replacing the missing data with a value (e.g., 90) yields a valid repair. However, deleting either the humidity feature or the New York entry does not constitute a valid repair despite eliminating the missing value.*

Table 1: Training set with missing value

	Temperature (F)	*Humidity (%)*	*Rain (1) or not (-1)*
Seattle	*65*	*80*	*1*
New York	*50*	*null*	*-1*

3 CERTAIN MODELS

Our initial step is to provide a formal definition of certain models that minimize training loss irrespective of the missing data's values.

Definition 2. *A model $\boldsymbol{w}^*$ is a certain model if:*

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

When a certain model exists, imputing missing data is unnecessary since this model is always optimal in model training for all repairs. Figure 1 demonstrates this, as a certain model exists in Figure 1a but not in Figure 1b. In unsupervised learning, certain models are defined in the same way as supervised learning but only considering the repairs for feature inputs: there are no label outputs in unsupervised learning.

In the case of a data set with missing values, the first challenge is determining *whether a certain model exists.* If a certain model exists, we can confidently ignore missing data and use this model for downstream ML. Therefore, the second challenge is *learning a certain model* if it exists.

Given Equation 1, a *baseline algorithm* to check for the existence of a certain model is: (1) learning models from all repairs one by one, and (2) a certain model exists if all repairs share at least one mutual optimal model. However, this baseline method can be incredibly slow due to a large number of repairs, so we aim to find a faster algorithm. In fact, it's challenging to develop an efficient algorithm that applies to all ML models because solving the optimization problem in Equation 1 is generally hard without strong assumptions. Therefore, we present efficient algorithms for specific ML models such as linear regression in Section 4 and SVM in Section 5. To simplify, we assume the missing values are only present in feature input, not label output, for the two ML models.

4 CERTAIN MODELS FOR LINEAR REGRESSION

The problem formulation for certain model $\mathbf{w}^*$ in linear regression is: $\forall \mathbf{X}^r \in \mathbf{X}^R$, $\mathbf{w}^* = \underset{\mathbf{w} \in \mathcal{W}}{\arg\min}\, ||\mathbf{X}^r\mathbf{w} - \mathbf{y}||_2^2$. Linear regression finds the best set of linear coefficients (i.e., $w_1^*, ..., w_d^*$) such that the linear combination of column vectors, $w_1^*\mathbf{x}_1 + ... + w_d^*\mathbf{x}_d$, has the shortest Euclidean distance to the label vector $\mathbf{y}$, i.e., the minimum training loss. Intuitively, a certain model exists when this Euclidean distance is independent of the columns $\mathbf{x}_j, j \in J$. Otherwise, for model $\mathbf{w}^*$ that is optimal to repair $\mathbf{X}^{r1}$, one can always find another repair $\mathbf{X}^{r2}$ that makes the training loss not minimum against $\mathbf{w}^*$ due to the dependency on $\mathbf{x}_j, j \in J$. To check if a certain model exists, we can use Theorem 4.1, which avoids the need to check all possible repairs. Denote a missing-value-free matrix $\mathbf{C}$ by the submatrix of $\mathbf{X}$ such that $\mathbf{C}$ consists of all the columns $\mathbf{x}_j, j \notin J$. The model $\mathbf{w}_C^*$ is learned from fitting $\mathbf{C}$ and $\mathbf{y}$ to linear regression, and $\mathbf{t}$ is the residue from this fitting, i.e., $\mathbf{t} = \mathbf{C}\mathbf{w}_C^* - \mathbf{y}$.

Theorem 4.1. *A certain model exists if and only if ;$\forall j \in J$, the two conditions are met: 1) for any $(i,j) \in MVX$, $t_i = 0$; 2) $\sum_{(i,j) \notin MVX} X_{ij} \cdot t_i = 0$.*

Theorem 4.1 states that a certain model exists for linear regression if and only if the residue vector $\mathbf{t}$ is orthogonal to all column vectors with missing values. In other words, if there is a certain model, then the columns with missing values can be safely ignored as they cannot contribute to a smaller training loss than the Euclidean norm of $\mathbf{t}$. Therefore, a more efficient algorithm than the baseline is available to check certain models without traversing over all repairs: 1) computing the residue vector $\mathbf{t}$ based on missing-value-free columns, and 2) checking the orthogonality between $\mathbf{t}$ and all missing columns. The proof for Theorems 4.1 and 5.1 and the related algorithms are available *here*).

Further, we can obtain the certain model when it exists by filling zero vectors into $\mathbf{w}^*_C$, which ensures that the column vectors with missing values are ignored by the zero linear coefficients.

5 CERTAIN MODELS FOR SVM

The hinge loss term in the loss function of SVM is $C\sum_{i=1}^{n} max\{0, 1-y_i\mathbf{w}^T\mathbf{e}_i\}$. For $i \in I$, the hinge loss becomes $C\max\{0, 1-y_i(w_1 \cdot X_{i1} + ... + w_d \cdot X_{id}\}$. Suppose $(i, j) \in MVX$, intuitively, a certain model exists when X^r_{ij} in any repair $\mathbf{X}^r \in \mathbf{X}^R$ does not affect the minimization of hinge loss. Given this, Theorem 5 offers a way of checking certain models. Denote a missing-value-free matrix $\mathbf{D}$ by another submatrix of $\mathbf{X}$ such that $\mathbf{D}$ consists of all the rows $\mathbf{e}_i, i \notin I$ from $\mathbf{X}$. Correspondingly, a subvector $\mathbf{y}_D$ is defined by consisting of all $y_i, i \notin I$. $\mathbf{w}^*_D$ is the SVM model learned from $\mathbf{D}$ and $\mathbf{y}_D$.

Theorem 5.1. *A certain model exists if and only if when the two conditions hold: 1)* $\forall j \in J, w^*_j = 0$, *and 2)* $\forall i \in I, y_i \sum_{j \notin J} w_j X_{ij} > 1$.

The theorem tells that a certain model for SVM exists if and only if when none of the training examples that have missing values can possibly be a support vector. Therefore, these training examples are *irrelevant or redundant* given other missing-value-free training examples. As a result, an efficient algorithm is also available for SVM to avoid traversing over all repairs: 1) learning $\mathbf{w}^*_D$ with missing-value-free training examples, and 2) checking the two conditions in Theorem 5.1 against $\mathbf{w}^*_D$.

If a certain model is determined to exist, $\mathbf{w}^*_D$ is exactly the certain model based on the proof for Theorem 5.1.

6 LESSONS LEARNED

The algorithms for linear regression and SVM are designed to determine if missing features or training examples are irrelevant or redundant when minimizing training loss, given non-missing data. In practice, data irrelevance and redundancy are often addressed after handling missing data because missing data is typically assumed to be significant. In the paper, our algorithms are focused on cases where missing features/training examples are irrelevant or redundant irrespective of the missing data's actual values.

When dealing with certain models in practical ML problems, three key issues can arise. Firstly, while certain models may allow for the omission of missing features or training examples, this approach can lead to overfitting, particularly in datasets with a high number of missing values. Secondly, checking certain models may not be feasible in practice due to the strict optimality requirements of all repairs. Thirdly, certain model algorithms rely on constraints in the hypothesis space, such as linearity in linear regression, making it difficult to efficiently verify certain models for flexible ML models like neural networks.

To overcome these challenges, it is crucial to develop a relaxed optimality condition and leverage non-missing values in missing features/training examples. Additionally, probabilistic models with repair sampling can facilitate the development of efficient algorithms for general ML models. Finally, given the importance of data cleaning in many scenarios, combining certain model algorithms with popular progressive data cleaning frameworks may prove effective in real-world datasets.

7 RELATED WORK

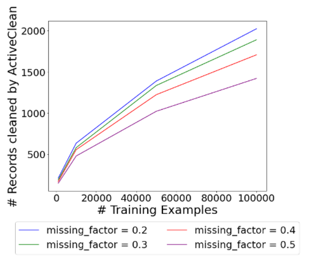
ActiveClean. ActiveClean prioritizes cleaning training examples with large model gradients and the training loss is incrementally minimized using SGD until convergence to the true minimum[3]. Its goal aligns with certain models that guarantee minimum training loss. However, ActiveClean does not always stop sampling and cleaning when a certain model exists. Here we compare the performance of our work with ActiveClean by considering the problem: if the data is dirty by missing values only and a certain model exists, learn an optimal model. In terms of generalization, ActiveClean only works for convex problems as required by SGD to converge to a global minimum. In contrast, the baseline algorithm in Section 3 checks and learns certain models for all optimization-based ML problems. Then, we divide the costs into cleaning and numerical operations. For the cleaning costs, the worst-case scenario in ActiveClean is to clean all random values that replace missing values at initialization since they are very likely wrong values. In comparison, the certain model method is completely cleaning-free. For the costs of numerical operations, ActiveClean first pays $O(T_{train})$ to learn an initial model from the random repair. Then, it takes $O(\frac{1}{\epsilon^2})$ iterations to converge as claimed in the paper. In our work, the checking and learning of certain models cost $O(|\mathbf{X}^R| * T_{train})$, which reflects the baseline algorithm in Section 3. If the problem is convex, faster algorithms may be available, e.g., $O(T_{train})$ for linear regression and SVM in this paper.

DLearn. Instead of imputing missing data, DLearn leverages the database constraints and learns a relational model without cleaning [5]. The cleaning-free method digests missing data by covering as many positive and as few negative examples as possible. Therefore, DLearn returns a model that is most likely optimal. However, the method is limited to relational learning.

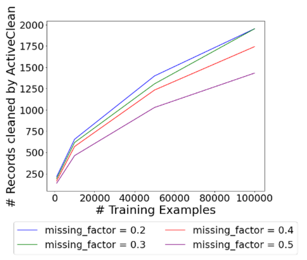
CPClean. CPClean is also a progressive cleaning framework but checks for unnecessary data cleaning based on the certain prediction of a model on the validation set [2]. However, this condition does not necessarily guarantee an optimal model, especially when the validation set is small or dirty. In comparison, our work focuses on the training stage and guarantees a minimum training loss.

8 EMPIRICAL EVALUATION

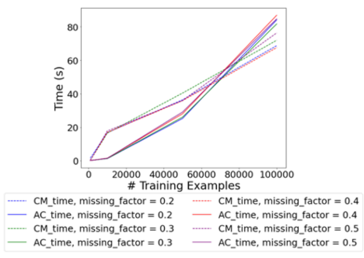
We demonstrate the ability of our algorithms to check certain models and save cleaning costs compared to the benchmark method, ActiveClean. Additionally, we analyze the time cost of running these model algorithms on various training set sizes. We evaluated the performance of these methods on Linear Regression and SVM, using synthetic datasets.



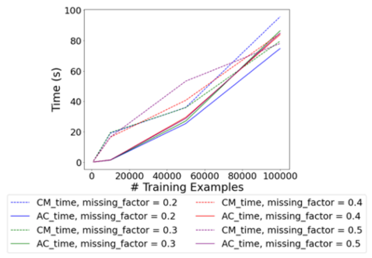
(a)



(b)



(c)



(d)

Figure 2: # Records Cleaned vs # Training Samples for Linear Regression (2a) & SVM (2b) Run-time cost vs # Training Samples for Linear Regression (2c) & SVM (2d)

8.1 Setup

To ensure strict optimality for all repairs in certain machine learning models, we created synthetic data that satisfied the optimality conditions for both linear regression and SVM problems. To simulate real-world datasets, we carefully selected parameters for data generation. The resulting dataset had between 1000–100, 000 records, and the dimension of the feature vector was fixed to 5, 000 to mimic real-world feature sizes. We also introduced missing values into the dataset based on the Missing Factor, which had values ranging from between 0.2–0.5. Finally, we used a fairly traditional 80%–20% Train-Test split for our experiments.

The data for both Linear Regression and SVM problems was generated through simulation. For Linear Regression, a linear combination of informative and non-informative features was utilized to predict the target value. The informative features contributed to the target while the non-informative features did not. For SVM, the data was drawn from a scaled uniform distribution with the number of samples and features specified by the parameters. The feature matrix and target vector were obtained by reflecting the features along all dimensions except one and negating the labels, which allowed for the model parameters corresponding to the selected features to be equal to zero. To ensure real world scenarios were accurately represented, we introduced missingness in the dataset by randomly imputing some values based on the missing factor. We implemented the models using Scikit-learn version 0.24.2 and compared our results to the code published by the authors of ActiveClean[4].

8.2 Saving cleaning cost

Given the input data, we learned certain models by implementing Algorithms for Linear Regression and SVM without performing any cleaning. In comparison, ActiveClean continued cleaning large amounts of data even though a certain model existed from the initialization. As missing factors or training examples increased, ActiveClean surprisingly performed less data cleaning before stopping. We can probably attribute it to ActiveClean providing less weightage to the feature with a lot of missing values as it wouldn't be affecting the target. We tested the performance of models learned from both methods in testing sets by using the R^2 score and Accuracy Score for Linear Regression and SVM respectively and found that both methods had a perfect score due to the synthetic dataset's special setup. While lacking real-world data set results, our method can learn a model with comparable performance to ActiveClean while saving on cleaning costs if a certain model exists.

8.3 Comparing computational costs

Our findings demonstrate that both methods have similar time costs across all testing parameters, particularly when a certain model already exists. Even when a certain model is absent, the computational overhead from checking for it is equivalent to running ActiveClean for the two ML problems. Additionally, as highlighted in the ActiveClean paper, the primary time cost stems from human intervention in the cleaning process. By checking for a certain model before implementing ActiveClean, significant time and resources can be saved. Therefore, our results strongly suggest that our approach can effectively optimize the cleaning process, while minimizing the need for human intervention.

9 CONCLUSION

In this paper, we have presented certain models as the condition to ignore missing data in ML. Efficient algorithms for checking the existence of and learning certain models are offered for linear regression and SVM problems. Through experiments, we have demonstrated the cleaning cost saving compared to a benchmark "cleaning for ML" method while not bringing much overhead to the running time cost.

REFERENCES

[1] Dong Deng, Raul Castro Fernandez, Ziawasch Abedjan, Sibo Wang, Michael Stonebraker, Ahmed K Elmagarmid, Ihab F Ilyas, Samuel Madden, Mourad Ouzzani, and Nan Tang. 2017. The Data Civilizer System.. In *Cidr*.

[2] Bojan Karlaš, Peng Li, Renzhi Wu, Nezihe Merve Gürel, Xu Chu, Wentao Wu, and Ce Zhang. 2020. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. *arXiv preprint arXiv:2005.05117* (2020).

[3] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. 2016. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment* 9, 12 (2016), 948–959.

[4] Krishnan, Sanjay and Wang, Jiannan and Wu, Eugene and Franklin, Michael J and Goldberg, Ken. 2018. Cleaning for Data Science. https://activeclean.github.io/

[5] Jose Picado, John Davis, Arash Termehchy, and Ga Young Lee. 2020. Learning over dirty data without cleaning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1301–1316.

[6] Stef Van Buuren. 2018. *Flexible imputation of missing data*. CRC press.

[7] Zhen, Cheng and Chabada, Amandeep Singh and Termehchy, Arash. 2023. When Can We Ignore Missing Data in Model Training. https://eecs.oregonstate.edu/~termehca/certaintrain.pdf