CertainPrep: Human-in-the-Loop On-Demand Imputations for Supervised Learning Data Prep

Cheng Zhen*
Oregon State University
Corvallis, Oregon
zhenc@oregonstate.edu

Nischal Aryal*
Oregon State University
Corvallis, Oregon
aryaln@oregonstate.edu

Arash Termehchy
Oregon State University
Corvallis, Oregon
termehca@oregonstate.edu

Alireza Aghasi
Oregon State University
Corvallis, Oregon
alireza.aghasi@oregonstate.edu

Saibalaji Neeli
Oregon State University
Corvallis, Oregon
neelis@oregonstate.edu

ABSTRACT

Real-world data often contains dirty data such as missing values, outliers, and inconsistencies. To train accurate models over real-world datasets, users need to spend a substantial amount of time and resources imputing and finding proper values for cleaning training examples. This paper demonstrates that it is possible to learn accurate models without data imputations for certain training data and machine learning models. We propose a unified approach for checking the necessity of data imputation to learn accurate models across various widely used machine learning paradigms, allowing users to define their requirements for a model's accuracy. Built upon this approach, our CertainPrep system checks this necessity efficiently with theoretical guarantees and returns accurate models in cases where imputation is unnecessary. When imputation is required, the system recommends minimal training examples to impute and involves users in trying different imputation methods and comparing them. Our demonstration emphasizes the time and manual effort savings from avoiding data imputation while delivering accurate machine learning models.

PVLDB Reference Format:
Cheng Zhen, Nischal Aryal, Arash Termehchy, Alireza Aghasi, and Saibalaji Neeli. CertainPrep: Human-in-the-Loop On-Demand Imputations for Supervised Learning Data Prep. PVLDB, XXX-XXX, 2024.

1 INTRODUCTION

The performance of a machine learning (ML) model relies substantially on the quality of its training data. Real-world training data often contain dirty data, such as missing values, outliers, and inconsistent data. One may train an ML model by ignoring the training examples with dirty data. This approach, however, may significantly reduce the accuracy of the resulting model as it may lose out on some useful examples.

*Both authors contributed equally to this research.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

To address the problem of training over dirty data, users usually replace each dirty data item with a value that is considered reasonable, i.e., data imputation, and train their models over the resulting *repaired data.* To accurately impute data, users often need to figure out the mechanisms for dirty data. For instance, to impute a missing value, it is crucial to determine if the value is missing completely at random or based on observed values of some features. Based on this mechanism, they build a (statistical) model for missing data and replace the missing values with some measurements defined over this model, e.g., mean. The aforementioned steps of finding a mechanism for dirty data and finding an accurate imputation method require a significant amount of time and manual effort. Surveys indicate that most users spend about 80% of their time on data preparation and repair [3].

Researchers have recently shown that one may learn accurate Datalog rules [2] and K-nearest neighbor classifier [1] over a training dataset without repairing dirty data. These methods check if dirty data influence the target model and, if not, return the model learned over the original training data. This approach may save significant time and effort spent repairing data.

However, it is not clear whether the methods above can be used to check the necessity of data repair for other ML models. Unlike learning Datalog rules or K-nearest neighbors, training popular ML models usually requires optimizing a continuous loss function. Additionally, these methods detect the necessity of data repair only for classification models and do not handle regression models. Furthermore, each of these methods can only handle a single ML model. Ideally, one would prefer a single system to learn over datasets with dirty data for multiple types of ML models.

In this paper, we demonstrate a system, CertainPrep, for learning accurate ML models with minimal data imputations. CertainPrep focuses on ML models that optimize loss functions over continuous spaces, which arguably contain the most popular ML models. Within the system, we formally define the necessity of data imputation for learning accurate models over dirty data. Generally, people need more data cleaning efforts to build a more accurate ML model while the need for the model's accuracy varies among applications. Therefore, users are invited to define their requirements for an ML model's accuracy. CertainPrep efficiently detects whether data imputation is needed to learn accurate models. If imputation is unnecessary, the system learns accurate models over the original training data. However, when imputation is required, CertainPrep suggests a minimal number of training examples to impute that help achieve accurate ML models.

2 BACKGROUND

In this section, we review terminology and notations in ML, as well as the dirty data that CertainPrep handles.

Supervised Learning. Given n training examples, a training set consists of feature input $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]^T$ and label output $\mathbf{y} = [y_1, ..., y_n]^T$. 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., $\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathcal{W}} L(f(\mathbf{X}, \mathbf{w}), \mathbf{y})$.

Dirty Data. CertainPrep covers three types of dirty data that require imputations. 1. Missing value: any x_{ij} is a missing value if it is unknown (marked by *null*). 2. Inconsistent data: for inconsistency, CertainPrep currently handles constraint violations over individual tuples in a table. 3. Outlier: a value x_{ij} is an outlier if it substantially deviates from the distribution of the corresponding feature. The identification of outliers relies on specific outlier detectors.

Example 2.1. *In Table 1, temperature and relative humidity are two features, and rainfall is a binary label. The relative humidity value for New York's example (in red) is unknown, and thus a missing value. The relative humidity value for London's example (in blue) is inconsistent by violating the domain constraint that relative humidity values are between 0 and 100. Finally, the temperature value in Seattle's example (in yellow) is an outlier because it deviates from the distribution of the temperature feature.*

Table 1: A training dataset for rain prediction

	Temperature(F)	Relative Humidity(%)	Rainfall
New York	75	*null*	-1
London	50	105	-1
Seattle	-100	80	1
Berlin	65	30	1

Repair. A repair $\mathbf{X}^r$ is a version of imputation where all dirty data from the original dataset $\mathbf{X}$ are imputed.

Example 2.2. *Below is a valid repair $\mathbf{X}^r$ achieved through three imputations. Imputing the humidity in New York with a specific value (e.g., 90) fixes the missing value. Imputing the humidity in London with a value adhering to the domain constraint (e.g., 80) resolves the inconsistency. Imputing the temperature in Seattle with a value within the distribution (e.g., 60) addresses the outlier. However, simply deleting the examples from New York, London, and Seattle, which eliminates the dirty data, does not constitute a repair, as it alters the dimension of* $\mathbf{X}$.

Set of possible repairs. The range of values that can be used to replace dirty data is large. Consequently, a large number of repairs may exist. We denote the set of all possible repairs as $\mathbf{X}^R$.

3 SYSTEM DESCRIPTION

In this section, we first introduce our certain and approximately certain model method used to check the necessity of data imputation. If data imputation is determined necessary, we provide an approach to suggest a minimal number of examples with dirty data to impute. Finally, we outline the architecture of CertainPrep built based on certain and approximately certain model methods.

3.1 Certain Models (CM)

Here, we formally define certain models that minimize training loss irrespective of how data are imputed.

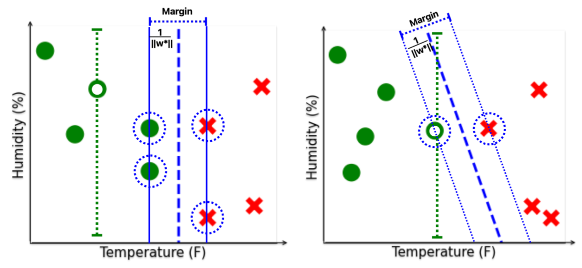
Definition 1. *(Certain Model) A model $\boldsymbol{w}^*$ is a certain model if:*

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

Where $\boldsymbol{X}^r$ is one possible repair, $\boldsymbol{X}^R$ is the set of all possible repairs and $L(f(\boldsymbol{X}^r, \boldsymbol{w}), \boldsymbol{y})$ is the loss function

Definition's intuition: Definition 1 says that if a model is optimal (minimizes the training loss) for all possible repairs, then this model is a certain model. Intuitively, imputation does not make a difference to model training when CMs exist and is thus unnecessary.

Example 3.1. *Consider the classification problem depicted in Figure 1, where the Support Vector Machine (SVM) model is used. The objective is to learn a linear boundary (blue rectangle) for rain prediction based on temperature and humidity features from various cities, which is consistent with the features and labels in Table 1. Figures 1a and 1b illustrate two sets of training examples with a missing humidity value. The green dashed line represents the range of possible imputations, with the empty circle denoting a possible imputation. In Figure 1a, different imputations lead to the same optimal model (decision boundary: blue dashed line), indicating the existence of a CM ($\boldsymbol{w}^*$). In other words, this CM $\boldsymbol{w}^*$ creates the widest margin between examples with different labels (i.e., minimizes the training loss) for all possible repairs. However, in Figure 1b, the optimal model varies depending on the chosen repair, suggesting that a CM does not exist.*



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

Figure 1: Data cleaning may not always be necessary

3.2 Approximately Certain Models (ACM)

The conditions for having certain models might be too restrictive for many datasets as it requires a single model to be optimal for all repairs of a dataset. In practice, however, users are usually satisfied with a model that is *sufficiently close* to optimality. Therefore, we leverage this fact and propose the concept of *approximately certain models*, which relaxes the conditions of certain models. An approximately certain model is within a given threshold from every optimal model for each repair of the input dataset. If there is an approximately certain model for a training task, users can confidently use the approximately certain model without any imputations.

Definition 2. *(Approximately Certain Model) Given a user-defined threshold $e \geq 0$, the model $\mathbf{w}^{\approx}$ is an approximately certain model if the following condition holds:*

$$\forall \mathbf{X}^r \in \mathbf{X}^R, L(f(\mathbf{X}^r, \mathbf{w}^{\approx}), \mathbf{y}) - \min_{\mathbf{w} \in \mathcal{W}} L(f(\mathbf{X}^r, \mathbf{w}), \mathbf{y}) \leq e \quad (2)$$

where $\boldsymbol{X}^r$ is a possible repair, $\boldsymbol{X}^R$ is the set of all possible repairs and $L(f(\boldsymbol{X}^r, \boldsymbol{w}), \boldsymbol{y})$ is the loss function.

Definition's intuition: Definition 2 ensures that the training losses of ACMs are close to the minimal training loss for all repairs by threshold e. When e is small, ACMs are approximately optimal for all repairs. Users can define the threshold e based on the need for the model's accuracy in their applications. For instance, a small e can be defined for a downstream application requiring high ML accuracy, although this decreases the likelihood of ACMs' existence. When ACMs exist based on the user-defined e, data imputation is unnecessary, allowing users to proceed with an ACM without significantly compromising the model's performance.

3.3 Checking and Learning CM and ACM

A *baseline algorithm* for checking and learning a CM or an ACM is: (1) learning *possible models* from all possible repairs one by one, (2) a CM exists if all repairs share at least one mutual optimal model, and (3) an ACM exists if there is a possible model that has sufficiently minimal training loss for all repairs. Here, the set of possible repairs is often large (Section 2). Therefore, *learning models from all repairs may be incredibly slow.*

Without materializing all repairs, CertainPrep provides efficient algorithms that check and learn certain models for linear regression, linear SVM, SVM with polynomial kernels, SVM with RBF kernels, and deep neural networks. The intuition of efficient algorithms lies in checking each example with dirty data and assessing whether any possible repair affects model training, i.e., the minimization of the training loss. For instance, the CMs for SVM are efficiently evaluated by Algorithm 1, which checks if any repair to an example with dirty data serves as a support vector. Proofs of the algorithms are available in our technical report[4].

Algorithm 1 Checking and learning CM for SVM

```
$Clean(\mathbf{x}) \longleftarrow$ set of clean examples that have no dirty data
$Dirty(\mathbf{x}) \longleftarrow$ set of examples that contain dirty data
$\mathbf{w}^{\diamond} \longleftarrow$ model trained with clean examples in $Clean(\mathbf{x})$
for $\mathbf{x}_i \in Dirty(\mathbf{x})$ do
    if $\exists \mathbf{x}_i^r$ that is a support vector with respect to $\mathbf{w}^{\diamond}$ then
        ▷ We check this condition without scanning all repairs [4]
        return "Certain models do not exist"
    end if
end for
return "A certain model $\mathbf{w}^{\diamond}$ exists"
```

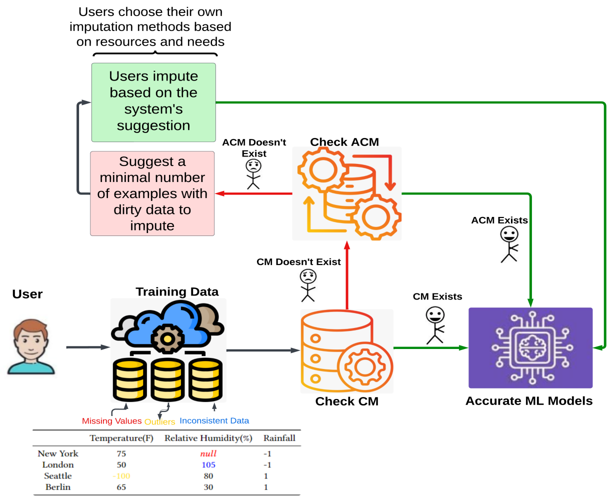
Benefits vs. Overhead in Checking CMs and ACMs: When CM or ACM exists, we save a significant amount of time and resources. Furthermore, these savings significantly grow as the number of datasets increases alongside the rapid expansion of the ML community utilizing these datasets for model training. In scenarios when neither CM nor ACM exists, checking for them incurs some computational overhead. Nonetheless, *paying for this overhead is worthwhile* for three reasons. First, substantial data cleaning savings are realized when CM or ACM exists as above-mentioned. Second, the time costs associated with checking CM and ACM are minimal as confirmed by our extensive experimental results [4]. Third, even in cases where neither CM nor ACM exists, our algorithm can select a subset of examples with dirty data to suggest for imputation (Section 3.4). This approach leads to cost savings in data cleaning, as it can avoid the need to impute all dirty data.

3.4 Suggesting Minimal Number of Imputations When Neither CM nor ACM exists

As above-mentioned, our CM checking algorithm involves scanning examples with dirty data, and identifying violations of CM conditions. In cases where neither a CM nor an ACM exists, our system returns a subset of examples with dirty data that violate the CM conditions. After imputing all the suggested examples, the repaired dataset satisfies all CM conditions, resulting in the existence of an accurate CM with the repaired dataset.

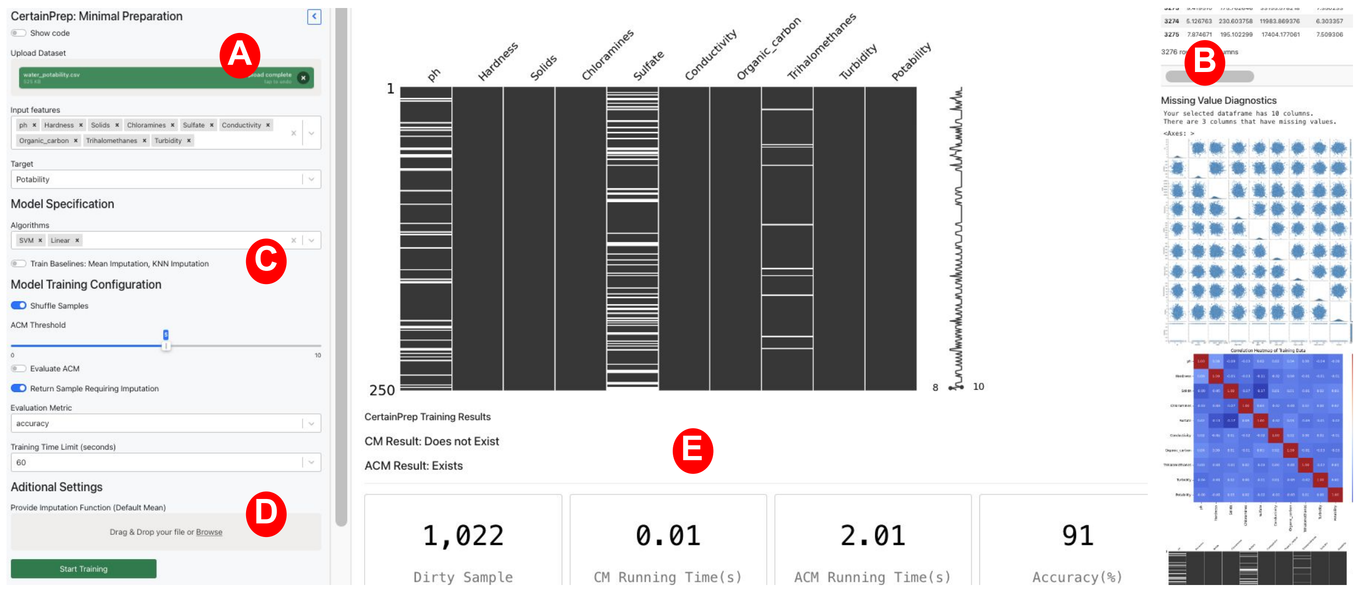
3.5 System Architecture

As illustrated in Figure 2, CertainPrep takes training data and identifies missing values, inconsistent data, and outliers. The system then evaluates the need for imputation by first checking the existence of CM. If a CM exists, CertainPrep provides a guaranteed accurate CM without any imputation. If CM does not exist, our system examines the existence of ACM based on the user's input for model accuracy requirements. When an ACM exists, users also benefit from an accurate ML model with zero imputation. In cases where imputation is determined necessary after the ACM checking, CertainPrep suggests a minimal number of examples with dirty data to impute. After users impute the suggested examples using their chosen method, CertainPrep returns an accurate model, completing the workflow.


Users choose their own imputation methods based on resources and needs
Users impute based on the system's suggestion
Suggest a minimal number of examples with dirty data to impute
ACM Doesn't Exist
Check ACM
ACM Exists
CM Doesn't Exist
User
Training Data
CM Exists
Check CM
Accurate ML Models
Missing Values
Outliers
Inconsistent Data

	Temperature(F)	Relative Humidity(%)	Rainfall
New York	75	*null*	-1
London	50	105	-1
Seattle	-100	80	1
Berlin	65	30	1

Figure 2: System Architecture for CertainPrep


CertainPrep: Minimal Preparation

Show code

Upload Dataset

water_potability.csv

Input features: ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_carbon, Trihalomethanes, Turbidity

Target: Potability

Model Specification

Algorithms: SVM, Linear

Train Baselines: Mean Imputation, KNN Imputation

Model Training Configuration

Shuffle Samples

ACM Threshold: 5

Evaluate ACM

Return Sample Requiring Imputation

Evaluation Metric: accuracy

Training Time Limit (seconds): 60

Aditional Settings

Provide Imputation Function (Default Mean)

Drag & Drop your file or Browse

3274	5.126763	230.603758	11983.869376	6.303357
3275	7.874671	195.102299	17404.177061	7.509306

Missing Value Diagnostics

```
Your selected dataframe has 10 columns.
There are 3 columns that have missing values.
<Axes: >
```

CertainPrep Training Results

CM Result: Does not Exist

ACM Result: Exists

1,022	0.01	2.01	91
Dirty Sample	CM Running Time(s)	ACM Running Time(s)	Accuracy(%)

Figure 3: System Interface for CertainPrep

4 DEMONSTRATION

4.1 Datasets

We selected real-world datasets that vary in domain, size, and the number of examples with dirty data. The datasets include the Water Potability dataset, which *classifies* water suitability for drinking based on freshwater sources. The Intel Sensor dataset involves *classifying* whether readings came from a particular sensor. The COVID dataset aims to predict critical staffing shortages in hospitals using *regression*. Finally, the NFL dataset uses play-by-play logs from US Football games to predict team scores through *regression*.

4.2 System Interface

To achieve a more interactive demonstration, we design an intuitive interface building on top of notebooks (Figure 3). The data-view (A) provides data import and automatically populates possible features and targets. Users can also manually modify input features and the target label. The data-exploration view (B) presents a sample of the dataset and summary visualizations. After performing an exploratory analysis of the data, the user may define the ML model training configuration. Model-configuration panel (C) allows users to select the ML model to train, specify if they also want to evaluate ACM, provide ACM threshold, and set a verbose output returning a dirty sample that needs attention. Additionally, the analyst can provide a custom cleaning function (D) that CertainPrep can incorporate into the data preparation pipeline when CMs do not exist. After the analyst initiates training, the result view (E) displays a visualization of dirty data distribution and model training results.

4.3 Demonstration Workflow

In our demonstration, we will highlight CertainPrep's overall efficacy on real-world datasets, as well as provide the user with hands-on experience working in the ML data preparation workflow.

4.3.1 Interactivity. The demonstration will include all permutations of real-world data preparation scenarios with CertainPrep. That is, (1) when CM exists (2) when CM does not exist but ACM exists (3) when neither CM nor ACM exists. We believe that involving the participants in these practical scenarios will help gauge the benefits of the CM/ACM framework (subsection 3.3).

To further encourage interactivity during the demonstration. We will present a few simplified datasets and models with artificial errors that are small enough to clean entirely in a few minutes. Participants will be able to pick a dataset to train an ML model for both classification and regression tasks. Additionally, participants can specify data cleaning functions other than our CM/ACM methods, such as those from the sklearn package. They can also choose to manually impute some examples and request the system to re-evaluate the imputed dataset. By comparing the performance of the CM/ACM method with other user-chosen data cleaning functions, we will clearly demonstrate the cost savings in data cleaning achieved by the CM/ACM approach.

4.3.2 Performance View. Once the model training task is completed, the system will present the comparative results against existing data cleaning methods. For instance, (a) examples with dirty data cleaned (b) total training time, and (c) test accuracy. We will also incorporate a real-time 'Leaderboard' of participants' data preparation time to demonstrate significant savings in data preparation efforts with CertainPrep. Finally, the demo attendees will get an insider view of the system and examine how CertainPrep can be easily plugged into existing ML workflows.

REFERENCES

[1] 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).

[2] 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.

[3] Steven Euijong Whang and Jae-Gil Lee. 2020. Data collection and quality challenges for deep learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3429–3432.

[4] Cheng Zhen, Nischal Aryal, Arash Termehchy, Alireza Aghasi, and Amandeep Singh Chabada. 2024. Certain and Approximately Certain Models for Statistical Learning. arXiv:2402.17926 [stat.ML]