IJDC | *Conference Paper*

Collaborative Data Cleaning Framework: a Pilot Case Study for Machine Learning Development

Nikolaus Nova Parulian
University of Illinois at Urbana Champaign

Bertram Ludäscher
University of Illinois at Urbana Champaign

Abstract

This study experiments with collaborative data cleaning, a pivotal phase in data preparation for both analysis and machine learning. We used a provenance Data Cleaning Model (DCM) for multi-user scenarios to track changes on a dataset and conduct comprehensive experiments that simulate multiple data curators working collaboratively on a dataset. Furthermore, we analyzed how different data-cleaning scenarios to improve quality metrics of completeness and correctness of a dataset can affect the downstream machine learning modeling performance.

Submitted 10 February 2024 ~ *Accepted* 22 February 2024

Correspondence should be addressed to Nikolas Parulian, Email: nnp2@illinois.edu

This paper was presented at the International Digital Curation Conference IDCC25, 19-21 February 2024

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

Introduction and Overview

Data cleaning is a critical step the data preparation and aims to ensure the accuracy, consistency, and integrity of data used for analysis or machine learning. However, data cleaning is time-consuming and labor-intensive, often requiring domain expertise and manual effort. With the increasing availability of complex datasets, *collaborative* data cleaning has emerged as a promising approach to leverage multiple data curators' collective knowledge and skills to clean data more efficiently and effectively.

Building upon our previous work on providing a provenance model for Collaborative Data Cleaning (CDCM) (Parulian et al., 2021), (Parulian and Ludäscher, 2022), in this work we describe the use case and an example of implementation of the model for a machine learning development. We aim to address the challenges and opportunities associated with collaborative data cleaning and investigate how different data-cleaning scenarios can impact the cleaned datasets and outcomes for the downstream tasks—increasing the transparency by recording provenance on each cleaning step and analyzing the effect on the data quality (DQ) metrics. To this end, we propose an experiment that involves applying different variations of data-cleaning steps to simulate multiple curators working together on cleaning a dataset. Our provenance model can also support reporting queries for multi-curator scenarios such as: *Who changed the values in the dataset? ; Who contributed to the data-cleaning workflow development?*

The findings from our experiment use case contribute to our understanding of practical use cases for collaborative data cleaning. The outcomes of our analysis shed light on how a collaborative data cleaning process can impact the quality and reliability of the cleaned data and how it can influence downstream data analysis tasks or modeling outcomes. This study aims to advance the use of provenance for *trust* and *transparency* in collaborative data cleaning and provide valuable insights for data curators working on machine learning development.

Background and Motivation

Machine learning pipelines consist of a series of steps that transform raw data into a model that can be used to make predictions, including general processes related to *Data Preparation*, *Model Development*, and *Model Deployment*. Data cleaning as part of data preparation is a critical step in this pipeline, as it involves identifying and correcting errors, inconsistencies, and missing values in the data, which can affect the model's performance. Machine learning models are only as good as the quality of the data they are trained on; thus, the quality of the data cleaning process is essential for the pipeline's success.

In a collaborative data cleaning scenario that involves multiple curators, data cleaning steps can be a subjective process, as different curators may have different opinions on what constitutes an error in the data and how to improve/clean it. There is growing interest in developing automated and collaborative data-cleaning methods to address these challenges. Automated data cleaning methods leverage machine learning algorithms to detect and correct errors in the data (Li et al., 2021) (Berti-Équille and Comignani, 2021), while collaborative data cleaning methods involve multiple individuals working together to clean the data (Musleh et al., 2022), (Bhardwaj et al., 2014). These methods can help to reduce the time and effort required for data cleaning, improve the quality and consistency of the cleaning process, and facilitate the reuse of cleaning workflows across different datasets (Carminé, 2021).

The purpose of this experimental simulation is to extend the use of Data Cleaning Provenance Model (DCM) (Parulian et al., 2021) for multiple curators and provide these contributions: Utilizing the Collaborative Data Cleaning Model (CDCM) to capture cell-level provenance and analyze changes in a multi-curator scenario; Utilizing CDCM to capture and analyze data-cleaning workflows and count the attributions from different curators; Provide

experimental analysis on effectiveness of collaborative data-cleaning workflows for machine learning.

Datasets and Machine Learning Use Case

The *Airbnb Chicago* listings dataset used in this analysis is based on data retrieved *Mar 19, 2023* and contains *7,625* rows.[1] Each row represents a listing that was available at that time. The listings are identified by a unique *id* and are represented by the *name*, which typically reflects the place name. A listing is hosted by a *host* who is identified by their *host_id* and *host_name*, and a host can have multiple listings. The location of a listing is specified by its *latitude* and *longitude* on the earth and its *neighbourhood* in the Chicago area. A listing can have a type specified by *room_type*. The listings can be rented for a minimum number of nights specified by *minimum_nights*. The Airbnb listings can be rented for a specified *price*. The *host* of the listing is responsible for determining this price. In addition, if a listing has been rented, it may have customer reviews summarized by the *number_of_reviews* field.

The goal of cleaning this dataset is to improve the data quality for downstream machine learning tasks for specifying listings, specifically for developing a predictive model for listing price recommendations based on the potential dependent features. We used a regression model based on Extreme Gradient Boosting (XGBoost) (Chen and Guestrin, 2016) to develop the price prediction model. XGBoost is a powerful ensemble model well-suited for handling complex datasets with high dimensionality. For this task, we will use a total of six variables: latitude, longitude, minimum_nights, number_of_reviews, neighbourhood, and room_type. Our initial model is developed using the original Airbnb dataset, which we refer to as the original_dataset, assuming that this dataset has been curated and at its highest quality.

Provenance Model and Experimental Framework

Data Cleaning Framework

We intentionally added DQ problems to the *Airbnb* dataset, as shown in Figure 1. This created a *perturbed_dataset* for the data-cleaning experiment. To evaluate the dataset's quality, an evaluation function has been applied based on specific tasks, such as applying formatting constraints for a Date column that requires a certain format and checking for numerical constraint violations for numerical columns. In this experimental scenario, the task of the data curators was to identify and characterize the data/values errors in the dataset and devise a way to clean the dataset by developing a data cleaning script/recipe or denoising function. To make the workflow *reusable*, we used algorithmic cleaning steps that the contributors of this project have prepared.

We added the following noise functions to the dataset: (1) Duplicate values, when the same data is recorded multiple times; (2) Out of vocabulary, when a value is recorded that is not part of the defined list of possible values. (3) Inconsistent naming, when the same concept is referred to by different names or abbreviations; (4) Data type violation, when a value does not meet a certain expectation of data type in the column, such as alphabetical values that are recorded on a column that requires a numerical value; (5) Inconsistent formatting, when a value is recorded in different formats, such as different date or time formats. The examples and details for affected columns can be seen in Appendix table 4.

[1] Airbnb. Get the data: Inside airbnb. insideairbnb.com/get-the-data.html, March 19, 2023.

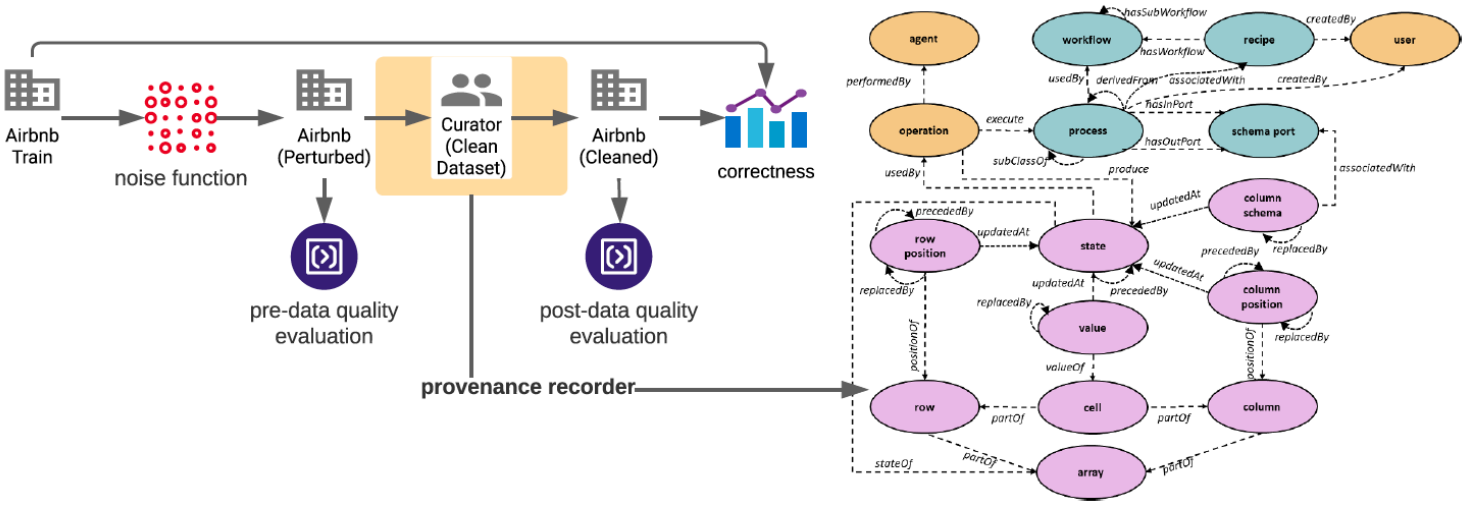


Figure 1. The dataset was subjected to experimental perturbation by introducing data quality (DQ) problems using a set of noise functions. This process involved intentionally modifying the values in the column to create variations or deviations from the original data. Data curators then cleaned the dataset based on the given collaboration scenarios. The changes made on the dataset (provenance) were recorded by an ad-hoc provenance recorder tool to fit the Collaborative Data Cleaning Model (Parulian and Ludäscher, 2022), enabling transparency via provenance queries.

For each cleaning scenario, as presented by the curator workflows, we evaluated the DQ by assessing two metrics *completeness* and *correctness* from the *perturbed* and *cleaned_dataset*. To evaluate *completeness*, we counted how many of the data rows can be fixed and retained for ML modeling. To evaluate *correctness*, we compared the *cleaned_dataset* values to the *original_dataset*, representing how well the data curator's cleaning approach on *perturbed_dataset* can recover the values back to their original values.

To facilitate data cleaning in a real-world scenario, we simulate the task of curators by providing scripts for data frame processing and data cleaning routines using *Python Pandas* [2]. We strategically divided the datasets based on data quality problems and captured diverse perspectives in the cleaning process. We accommodated our data cleaning conceptual collaboration framework Parulian and Ludäscher (2022) to divide the cleaning task by splitting the dataset based on their rows and columns (horizontal and vertical split). The experimental collaboration scenarios and environment for this data-cleaning workflow development step are accessible on GitHub repositories Parulian (2023). Appendix figure 4 shows an example of a data cleaning work template implemented on a Jupyter Notebook script.

Horizontal Split

In this collaboration scenario, a dataset is divided/split horizontally based on the rows (D_1 and D_2), and different curators C_1 and C_2 clean each subset. In our case, since we have divided the Chicago Airbnb dataset into two subsets, we implemented the collaboration scenario by assigning data-cleaning tasks for D_1 to C_1 and data-cleaning tasks for D_2 to C_2. By dividing the dataset into two subsets, the curators could work on cleaning the data more efficiently and become experts in the data quality and data cleaning issues specific to their assigned subset. Once the two subsets are evaluated and cleaned, they will be merged through a union operation to create a *cleaned_dataset* for the ML modeling.

[2] Python Pandas, https://pandas.pydata.org/pandas-docs/version/1.5/index.html, November 22, 2022.

Vertical Split

In this collaboration scenario, continuing the cleaning process after the horizontal split, the task of cleaning dataset D_2 is divided based on the columns. After evaluation, we found that C_2 only addressed some data quality problems. Thus, we introduced a new curator C_3 to handle the remaining problems that C_2 failed to address. In real-world scenarios, this can happen when a data curator lacks information or knowledge to tackle certain data quality issues or when possible bugs in the data-cleaning workflow need to be reviewed by another data curator. Since both curators worked on individual columns, the results from C_3 will be combined with C_2's results by selecting or projecting the relevant columns.

Workflow Merging

In addition to merging each cleaned data subset produced by the curators, the data-cleaning workflows developed by each curator can also be merged to create a comprehensive recipe or script for cleaning similar datasets for future reusability. However, merging these workflows requires careful consideration to ensure no conflicts or inconsistencies between the individual steps taken by each curator. Any inconsistencies or overlaps in the workflow steps must be resolved to ensure the final workflow is cohesive and effective. Additionally, it is important to ensure that the data types and formats used in the workflows are consistent across all steps to avoid any potential errors or conflicts.

By merging the cleaning workflows manually and considering the evaluation metrics, we can create a high-quality dataset that is well-suited for downstream tasks and provides valuable analysis. The findings suggest a significant correlation between different levels of DQ improvement due to the collaborative cleaning workflows to the ML prediction performance.

Figure 2 illustrates the overall workflow of collaboration tasks, highlighting how different curators can be assigned specific responsibilities based on horizontal or vertical data division. However, for the purpose of this study, we will simulate the curators' cleaning actions by executing data-cleaning workflows with different approaches (scripts/algorithms) to represent variations in cleaning the datasets. This will help us analyze the effectiveness of the data-cleaning steps and evaluate their performance by capturing and analyzing the changes made by different scenarios.

Data Cleaning Development Results and Analysis

This section examines the various data-cleaning processes and workflows utilized in the vertical and horizontal collaboration scenario. Each data-cleaning implementation is abstracted as an algorithm or programming script, with a focus on merging the resulting datasets through either concatenation for horizontal collaboration or column projection/ selection for vertical collaboration. We explain these workflows in detail, providing insights into the hypothetical collaboration scenarios. The resulting datasets and their provenance are then analyzed and compared using CDCM provenance model. The variation of the resulting datasets is obtained by executing and replicating different data-cleaning workflows C_1, C_2, and C_3, where each simulates or represents different curators.

Horizontal Split

Table 1 presents the outcomes of the vertical collaboration between *Curator 1 (C_1)* and *Curator 2 (C_2)* as recorded in the CDCM provenance model. The changes made to the dataset were tracked using an ad-hoc provenance recording mechanism specifically built for *Python Pandas Dataframe* for this experimental purpose. This reporting mechanism demonstrates the utility of the CDCM provenance model in tracking changes between different data-cleaning steps, user scenarios, and comparing values. In this experiment, since we can track the data from the original, perturbed dataset, and cleaned dataset, we use the recorded provenance information in

CDCM to compare each change with the ground truth original dataset to measure the completeness and correctness of the cleaned dataset.

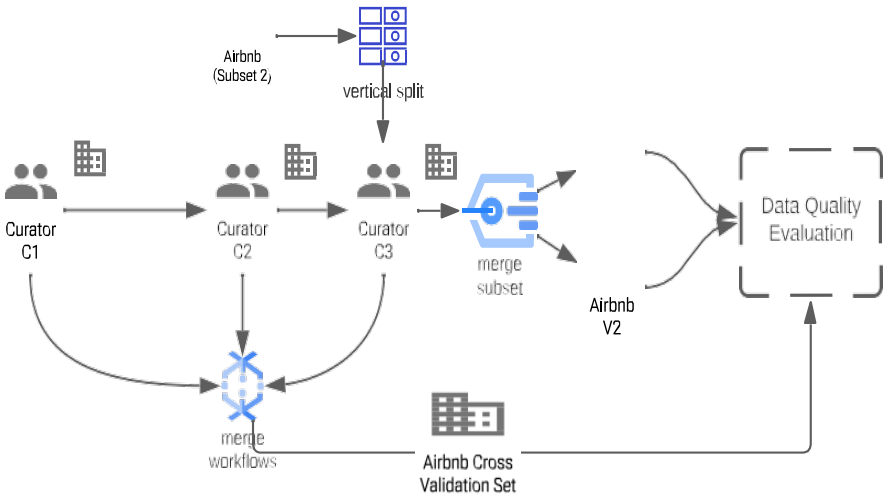


Figure 2. The experiment for a collaboration framework that involved both vertical and horizontal splits among the curators. In the horizontal split, the dataset is divided into subsets based on rows, and each subset is assigned to a specific curator for cleaning. Curators C_1 and C_2 will collaborate on cleaning different subsets of the dataset based on rows. Additionally, a vertical split is also implemented based on columns, where specific columns are assigned to different curators for partial cleaning. Curators C_2 and C_3 will collaborate on cleaning a subset of columns within the same dataset.

To merge the dataset based on the results of this horizontal collaboration, we can concatenate the dataset results from curators C_1 and C_2. As the columns were pre-determined, there should be no conflicting or missing columns when concatenating the results. Since the datasets were divided by rows, the two results represent different entities in the original dataset. The resulting concatenated dataset will be a new version named Cleaned Horizontal, which indicates the results of the horizontal collaboration.

Vertical Split

From the first iteration on horizontal collaboration, we can observe a scenario where curator C_2's data-cleaning workflows failed to clean the Latitude, Longitude, and Room type columns properly. As an example of a vertical split scenario on the second iteration, we hypothetically assigned the same task to clean those columns to curator C_3 by providing alternative methods in the cleaning scripts.

Table 1 shows the improvement of the performance of the resulting data cleaning after applying the approaches by curator C_3 on latitude, longitude, and room_type columns. Overall, the C_3 data cleaning workflows can recover all the noisy values introduced on the targeted columns, including Noise Type 2 missing '.' for latitude, longitude, and out of vocabulary problem for room_type column. This also suggests that having evaluation criteria can help to detect mistakes early and provide feedback to improve the workflow. It is important to have a robust and iterative approach to data cleaning, where the cleaning process can be improved based on feedback and evaluation results.

To merge the datasets based on the results of vertical collaboration, we can use a process called column projection. This involves selecting the columns that require cleaning and replacing the values in those columns with the corrected values provided by curators C_3 to C_2's dataset. In this case, we will be focusing on the latitude, longitude, and room_type columns of curator C_2 and replacing them with the corrected values from curator C_3. This results in a new version of the cleaned dataset, which we named Cleaned Vertical to indicate that it incorporates the results of the vertical collaboration.

Table 1. Report for changes based on the hypothetical data cleaning workflows by curator C_1,C_2, and C_3 on Dataset 1 and Dataset 2 respectively based on the provenance information as recorded in the CDCM. Each dataset has different data-quality problems that require different algorithms to simulate different curators. The curator can change the dataset by changing the values on the respective column or marking for deletion, resulting in Completeness (Com) and Correctness (Cor) metrics.

Column	D1 #Noise	C1 (D1)		D2 #Noise	C2 (D2)		C3 (D2)	
		Com	Cor		Com	Cor	Com	Cor
Neighbourhood	295	294	294	341	341	341	0	0
Latitude	73	73	73	85	0	0	85	85
Longitude	74	74	74	86	0	0	86	86
Minimum Nights	22	22	22	31	30	30	0	0
Number of Reviews	39	39	39	45	43	43	0	0
Room Type	178	178	178	192	0	0	192	192

Comparison on Downstream Task

The data curation process resulted in two versions of datasets: *Cleaned Horizontal* and *Cleaned Vertical*. The former is the result of merging the data cleaning results from curators C_1 and C_2 through concatenation. The latter is the result of a vertical merge of the data cleaning results from curators C_2 and C_3 on selected columns, which were then concatenated with the results from C_1. We have also observed that from our comparison analysis above, the results from C_1 did not have any significant issues that required different treatments and could, therefore, be retained for downstream tasks.

Table 2. Performance comparison on downstream task

Dataset	# Records (Train)	RMSE (Testing)
Original Dataset	3,050	212.58
Baseline - Remove all noise (BR)	1,762	362.05
Baseline - Average (BA)	3,050	207.41
Cleaned Horizontal (C1 + C2)	2,846	214.41
Cleaned Vertical (C1 + C2 + C3)	3,038	212.54

Table 2 provides a comparison of different datasets resulting from various data cleaning treatments. The performance of these datasets is measured based on two metrics: the number of

records in the training data and the root mean squared error (RMSE) over the testing data. The original dataset, which contains all the information without added noise, had 3,050 records, resulting in an RMSE of 212.58 for price prediction. The baseline removal approach, which removed all noise, resulted in a reduced number of records in the training data (1,762) and a higher RMSE of 362.05, indicating worse performance than the original dataset. The baseline average approach, which assigns default values (imputation) to the perturbed cells by performing statistical averaging, surprisingly performed better with an RMSE of 207.41. However, these results may not be generalizable since the analysis was conducted using partial data points. Further investigation is needed, especially in the context of workflow reusability analysis that will be discussed next, to validate these workflows' effectiveness across different noise intensities. Nevertheless, these findings suggest that when training a model on a noisy dataset, ensuring completeness by replacing missing values with reasonable defaults is crucial for achieving a performance that is closer to the ground truth.

The two cleaned datasets from the vertical and horizontal collaborations yielded promising results. The cleaned horizontal dataset contained 2,846 records of training data, which resulted in an RMSE of 214.41. In comparison, the cleaned vertical dataset contained almost all the dataset with 3,038 records and an even better RMSE performance of 212.54, similar to the performance on the original dataset. These results suggest that with the additional and continuous cleaning by C_3, the vertical collaboration approach provided a better outcome compared to the curator C_1 and C_2 horizontal approach, as it resulted in a higher number of records and a slightly better RMSE. As we can see from the table, both cleaned-curated datasets resulted in a reduction in the number of records compared to the original dataset. Still, the removal was relatively small compared to the baseline-removal approach. These results indicate that our data-cleaning workflows can positively or negatively impact the downstream task performance compared to the original dataset. Overall, the results of this study demonstrate the importance and effectiveness of collaboration in data curation processes.

Merging Workflow and Benchmarking for ML Development

In the previous section, we demonstrated that dividing and conquering a dataset horizontally or vertically can be effective for addressing specific problems. However, this approach has a disadvantage in terms of reusability. While we can replicate the data-cleaning workflow for different datasets, certain columns may have unique issues that require different cleaning strategies. Even though the script and recipes are available for replication or re-execution, blindly applying a single strategy may not comprehensively clean the data, particularly when problems are mixed within the dataset.

To holistically clean a dataset, we can merge different workflow scenarios that can help build up a complete strategy for cleaning the dataset. This involves applying different strategies to the entire dataset and testing how these different workflows perform on the overall dataset. We can combine the different cleaning strategies, such presented on horizontal and vertical cleaning, into a more comprehensive approach. This would involve applying different strategies to the entire dataset to identify and address all issues that may be present. By doing this, we can create a more complete and accurate dataset that can be used for a variety of downstream tasks.

One effective approach to combining different data cleaning processes and optimizing their outcomes without disrupting other steps is introducing new processes specifically designed to detect and handle identified data quality problems. These processes can incorporate flagging and signaling mechanisms that indicate which entities are affected by the specific problem. By encapsulating the detection method within each process, we can ensure that only the problematic entities are targeted for changes, minimizing unintended consequences for other data-cleaning steps. Furthermore, we can implement a safety mechanism by delaying the flagging for deletion until later in the data-cleaning workflow. This ensures that entities are

flagged for removal only when no other actions can be performed to address the identified problem. This strategy helps prioritize the most effective data-cleaning actions while avoiding unnecessary deletion of data that may still be salvageable. Appendix algorithm 1 provides an illustration of how we can incorporate this strategy to reuse and combine existing workflows clean_latitude_c1 and clean_latitude_c3. By integrating this strategy into the data cleaning workflow, we can ensure that the processes adhere to the rules and instructions for identifying and addressing data quality violations. Overall, this approach enables effective combination and optimization of data cleaning processes while minimizing the risk of unintended consequences. It promotes a targeted and controlled approach to data cleaning, ensuring that the most critical issues are addressed while preserving valuable data.

Table 3. Performance comparison for workflows reusability on the variation of datasets with different intensities of Data Quality Problems

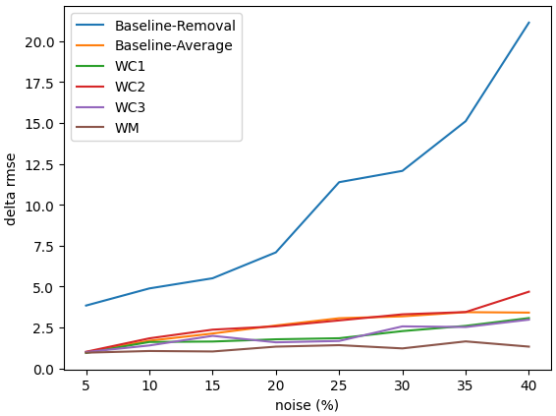
	Delta RMSE (mean ± standard deviation) over Data Quality Problems Intensity							
Workflow	**5%**	**10%**	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**
BR	3.84±1.53	4.89±1.56	5.51±1.88	7.09±1.91	11.38±5.46	12.07±5.86	15.11±7.11	21.13±8.61
BA	0.99±0.58	1.69±0.73	2.14±1.16	2.63±1.48	3.07±1.46	3.18±1.32	3.44±1.16	3.41±1.26
WC1	0.95±0.92	1.62±0.91	1.65±0.73	1.78±1.21	1.85±1.26	2.28±0.97	2.6±1.01	3.08±2.1
WC2	1.02±0.85	1.84±0.89	2.37±1.2	2.57±1.52	2.94±1.81	3.31±1.48	3.45±1.37	4.69±1.35
WC3	1.00±0.66	1.41±0.7	1.99±1	1.60±1.08	1.68±1.36	2.57±0.86	2.53±1.5	2.98±1.2
WM	0.96±0.65	1.07±0.87	1.04±0.68	1.33±0.89	1.42±0.65	1.23±0.66	1.65±0.76	1.33±0.88

We have merged the recipes based on the best performers from our analysis denoted as workflow WM. As a disclaimer, this merging was done manually, and we have full control over the decisions on which workflow to pick based on quality assessment. For the Neighbourhood column, we found that C_1 performs the best in terms of precision. However, it is also noteworthy that C_2 incorporates a threshold mechanism to gatekeep the changes, making sense as we may encounter various noisy characters. Therefore, we chose C_2 for cleaning the Neighbourhood column. Since the Latitude and Longitude columns have different data quality problems, we aimed to complement the workflows to address each problem effectively. As a result, we selected C_1 and C_3 workflows for merging, as they performed the best for their respective problems. Similarly, for the Room Type column, we chose to merge C_1 and C_3 workflows to complement each other in cleaning the data. In the case of the Number of Reviews and Minimum Nights columns, the approach used by C_1 was found to be better than C_2, so we selected C_1 workflow for these columns.

The workflows developed in this data-cleaning use case can be applied to different datasets with similar schemas and data quality issues, making them reusable. We define reusability as the ability to apply the same data-cleaning workflow on a new dataset with the same identified data quality issues. To evaluate the workflow's reusability and its performance on downstream prediction tasks, we created a new set of datasets consisting of 10 variations or subsets of training and testing data. To introduce noise, we applied various intensities of noise on the training data, ranging from 5% to 40% on each noise type randomly distributed over each dataset. Next, we applied the workflow developed in the previous stage to clean the data. The cleaned dataset versions for each workflow were processed through the same machine learning model and evaluated on each test data.

As a result of using these strategies, we annotate the new workflow in the CDCM to get the list of attributions for the new data-cleaning workflow *WM* as shown in the reporting result in Appendix table 5. As shown in the table, using the provenance in a level of workflow process can help take into account the contributions of each curator. For example, the report shows that although the new process *clean_latitude_c1_c3* was created by **Integrator 1**, it was developed based on *clean_latitude_c1* by curator $\mathbf{C_1}$ and *clean_latitude_c3* by curator $\mathbf{C_3}$.

Figure 3 illustrates the average performance comparison of different data-cleaning workflows on various datasets and noise intensity. The comparison is based on the delta root mean squared error (RMSE), which measures how different the performance of the cleaned datasets is compared to the original (ground truth) datasets when used for training a machine learning model. From the ten sets of cross-validation ground truth datasets, we get the average RMSE metrics of 117.41. The results indicate that the Baseline-Removal BR method, which involves excluding any values in the training data, performs poorly as data quality problems increase. This is expected because removing violated entities from the training data can result in incompleteness and hinder important information from the machine learning model.



Workflow	Coefficient	Std. Err	R^2
BR	0.453	0.022	0.839
BA	0.105	0.005	0.822
WC1	0.080	0.005	0.737
WC2	0.116	0.006	0.825
WC3	0.080	0.005	0.765
WM	0.047	0.004	0.662

Correlation	Completeness	Correctness
Delta RMSE	-0.671	-0.446

Figure 3. Data-cleaning workflow delta-RMSE comparison of price prediction model (ground truth) against cross-validation datasets on variation noise intensities. As the noise intensity increases, the impact on the model's performance becomes more evident. The graph demonstrates that arbitrary cleaning workflows, such as removing "dirty data" or assigning default values, lead to higher discrepancies to the ground truth performance and lower performance (higher *RMSE*) in general compared to the workflow curators' approach.

The Baseline-Average BA treatment represents a workflow where the missing values are imputed with the average value of the respective column. This treatment performs better than the Baseline-Removal workflow, as it maintains the completeness of the dataset, but its performance also degrades as the intensity of data quality problems increases. This suggests that simply imputing values without considering the nature of the data quality problem might not be an optimal long-term solution. Inferred values can potentially hide underlying issues in the dataset, and thus, it is important to validate and ensure the correctness of the imputed values. The performance of this workflow is similar to the workflow with partial treatment performed by WC2, where Curator C_2 does not correct any of the longitudinal data quality problems.

Workflow WC1 and WC3, which represent partial solutions for data quality problems, perform better than the Baseline-Average treatment. This indicates that even with missing or uncovered data quality problems, developing a partial solution can make the performance closer to the ground truth compared to inferring values without validation as performed in Baseline Average workflow. This highlights the importance of manually assessing data quality and validating entities to ensure they are as similar to the ground truth as possible. As expected, the merged workflow WM, which represents a combination of WC1 and WC3, performs the best overall since the merged workflow can recover most of the perturbed values to the original values. This performance comparison reinforces the importance of carefully merging data-cleaning workflows to achieve the best possible performance for downstream tasks in different intensities of data quality problems.

The scripts to replicate this experiment, including dataset generation, data-cleaning workflows, provenance recording and reporting, and downstream model analysis, can be found in the GitHub repository (Parulian, 2023).

Conclusion

We have provided a practical example, experimental simulation for collaborative data- cleaning to train a machine learning model. The study began by identifying the dataset used, which was the Airbnb listings dataset. Our objective was to develop a price prediction model based on this dataset. We artificially introduced noise to the original dataset to simulate dirty data. We simulated the task of collaborative data cleaning by introducing different workflows with various predefined algorithms. To facilitate the collaboration, we divided the dataset into subsets with different data-quality problems to justify different collaboration scenarios and practiced horizontal and vertical splitting as the frameworks for collaboration.

We have captured the data-cleaning provenance using an ad-hoc provenance recorder for Pandas Data Frame that tracks cell changes, workflows, and user information to the CDCM model. Having the provenance information recorded in cell-level changes allowed us to report and compare the results of different cleaned datasets and measure the completeness and correctness of each data-cleaning task. We have provided the reports and queries for each curator related to the cell-level changes and data-cleaning workflow. We also showcased the attribution query that can be useful in providing attribution to the development of a data-cleaning workflow.

We have used completeness and correctness data quality metrics to evaluate the effectiveness of the data-cleaning approaches. These metrics enabled us to assess the extent to which the curated datasets retained the original data while correcting any errors or inconsistencies. Furthermore, we analyzed the variations in the cleaned datasets to measure their impact on the performance of the price prediction model. We gained insights into how different data-cleaning strategies affected predictive accuracy by comparing the model's performance across these dataset variations. In addition to the cleaned datasets, we demonstrated how the individual data-cleaning assignments could be merged to form a comprehensive and reusable data-cleaning workflow solution that provides queries for attributions based on the recorded provenance. This solution could be applied to cross-validation datasets, which we used for analyzing the consistency and effectiveness of different data-cleaning workflows over various intensities of data quality issues.

Despite the strengths of our study, several limitations need to be acknowledged. Firstly, we conducted a controlled experiment based on artificial noise, which might not accurately represent real-world scenarios where data quality problems can be more complex and difficult to identify. Secondly, the data cleaning steps performed to simulate different curators were based on pre-defined functions, which might limit the scope of manual data curation methods that often exist in real-world data. While our approach ensures replicability and reusability, further investigation often is required to assess the validity of the values in a real-world scenario. Finally, the provenance information harvested in our study was based on ad-hoc scripts, which might not be the most efficient or reliable way to gather information on the execution and provenance of the workflow. While our approach allowed us to collect the required information, more standardized and automated methods of collecting provenance could be considered in the future.

Acknowledgements

Thank you to the Illinois Data and Knowledge Systems research group (the University of Illinois Urbana-Champaign): Lan Li, Yilin Xia, and Meng Li for contributing to the piloting use cases and providing different variations on data cleaning workflows.

References

Berti-Équille, L., & Comignani, U. (2021, January). Explaining automated data cleaning with cleanex. In *IJCAI-PRICAI 2020 Workshop on Explainable Artificial Intelligence (XAI)*. https://hal.science/hal-03066026/

Bhardwaj, A., Bhattacherjee, S., Chavan, A., Deshpande, A., Elmore, A. J., Madden, S., & Parameswaran, A. G. (2014). Datahub: Collaborative data science & dataset version management at scale. *arXiv preprint arXiv:1409.0798*. https://doi.org/10.48550/arXiv.1409.0798

Carminé, R. L. A. (2021). Trudata: A collaborative platform for data cleaning.URL https://repositorio-aberto.up.pt/ bitstream/10216/135717/2/488201.pdf

Chen T., and Guestrin, C. X. (2016). A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-94)., New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. URL https://doi.acm.org/10.1145/2939672.2939785

Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., & Zhang, C. (2021, April). Cleanml: A study for evaluating the impact of data cleaning on ml classification tasks. In 2021 IEEE 37th International Conference on Data Engineering (ICDE) (pp. 13-24). IEEE.

Musleh, M., Ouzzani, M., Tang, N., & Doan, A. (2020, June). Coclean: Collaborative data cleaning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (pp. 2757-2760). ACM. ISBN 978-1-4503-6735-6. doi: 10.1145/3318464.3384698. URL https://dl.acm.org/ doi/10.1145/3318464.3384698

Parulian, N. Collaboration framework and experiment repository, 2023. URL https://github.com/nikolausn/collab-use-case.git

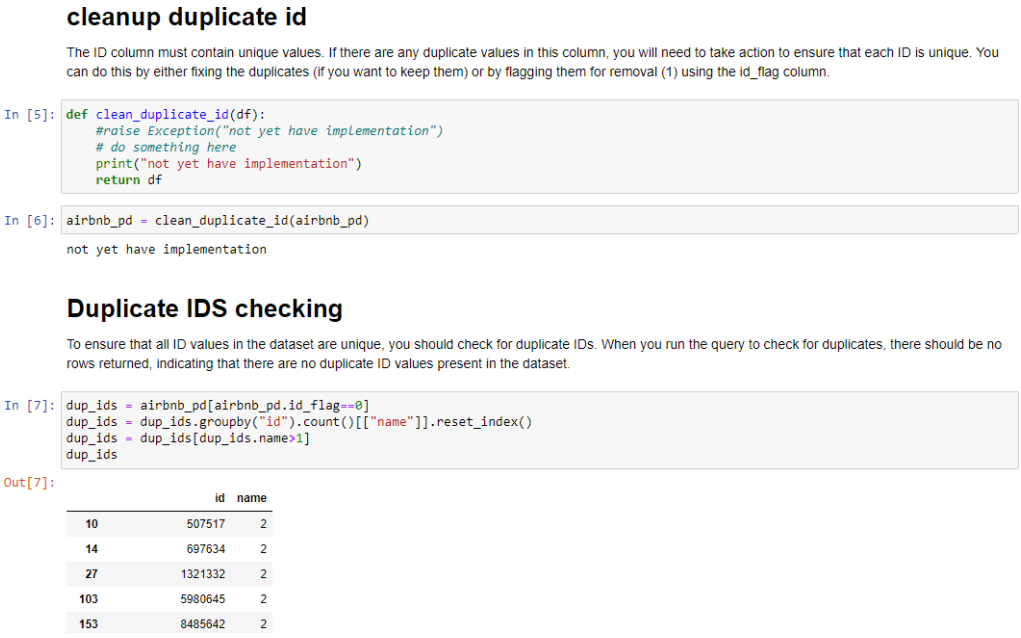
Parulian, N. and Ludäscher, B. (2022) Conceptual model and framework for collaborative data cleaning. In International Digital Curation Conference, 2022. URL https://doi.org/10.5281/zenodo.6781134

Parulian, N. N., McPhillips, T. M., & Ludäscher, B. (2021). A model and system for querying provenance from data cleaning workflows. In *Provenance and Annotation of Data and Processes: 8th and 9th International Provenance and Annotation Workshop, IPAW 2020+ IPAW 2021, Virtual Event, July 19–22, 2021, Proceedings 8* (pp. 183-197). Springer International Publishing.URL https://dx.doi.org/10.1007/978-3-030-80960-7_11

Appendix

Data-cleaning framework template example

An example template prompt for a data-cleaning task involves cleaning duplicate IDs. The data cleaning task consists of three sections: a short explanation of what needs to be done, a function that needs to be implemented by the curators, and the execution of the cleaning process. Additionally, a data quality check is performed to evaluate the current state of the dataset based on an evaluation function.



cleanup duplicate id

The ID column must contain unique values. If there are any duplicate values in this column, you will need to take action to ensure that each ID is unique. You can do this by either fixing the duplicates (if you want to keep them) or by flagging them for removal (1) using the id_flag column.

In [5]:
```
def clean_duplicate_id(df):
    #raise Exception("not yet have implementation")
    # do something here
    print("not yet have implementation")
    return df
```

In [6]:
```
airbnb_pd = clean_duplicate_id(airbnb_pd)
```

```
not yet have implementation
```

Duplicate IDS checking

To ensure that all ID values in the dataset are unique, you should check for duplicate IDs. When you run the query to check for duplicates, there should be no rows returned, indicating that there are no duplicate ID values present in the dataset.

In [7]:
```
dup_ids = airbnb_pd[airbnb_pd.id_flag==0]
dup_ids = dup_ids.groupby("id").count()[["name"]].reset_index()
dup_ids = dup_ids[dup_ids.name>1]
dup_ids
```

Out[7]:

	id	name
10	507517	2
14	697634	2
27	1321332	2
103	5980645	2
153	8485642	2

Figure 4

Data quality problems and examples

Examples of the data quality issues and artificially generated noises (dirty data) for each subset of the Airbnb Chicago listings dataset.

Table 4: Example data quality problems

Column	Type of Noise	Dataset	Original Value	Perturbed Value
Neighbourhood	Out of Vocabulary (Replace characters)	Dataset 1	Lake View	Lhke eieu
			Lincoln Park	LgScoLn sark
	Out of Vocabulary	Dataset 2	Near West Side Avondale	LNeaEr West aSide huAvongdale
Latitude	Wrong data type (Added characters)	Dataset 1	41.92748	E41.9A2748
			41.93203	4L1.9320u3
	Out of Range Numeric	Dataset 2	41.8167	418167
			41.8945286	418945286
Longitude	Wrong data type (Added characters)	Dataset 1	-87.7005	-8B7.7T005
			-87.68878	-87S.68A87p8
	Out of Range Numeric	Dataset 2	-87.6526915	-876526915
			-87.66632	-8766632
Room Type	Out of Vocabulary	Dataset 1	Entire home/apt	Entire home
	Out of Vocabulary	Dataset 2	Entire home/apt	Apartment
Minimum Nights	Wrong data type	Dataset 1/2	32	e32
Number of Reviews	Wrong data type	Dataset 1/2	74	7i4

Merging workflow

Example for merging existing workflows for Latitude column.

Algorithm 1 clean_latitude_c1_c3

Require: df , $df\ .latitude$, $df\ .latitude\_flag$ ◁ **data frame** as an input, required columns latitude, latitude flag

$df\ ErrorType1 \leftarrow detectErrorType1(df)$ ◁ function to detect and filter non-numeric data type error on Latitude column

$df \leftarrow clean\_latitude\_c1(df, df\ ErrorType1)$ ◁ apply clean_latitude_c1 to the dataset identified to have data type error

$df\ ErrorType2 \leftarrow detectErrorType2(df)$ ◁ function to detect and filter out of range violation on Latitude column

$df \leftarrow clean\_latitude\_c3(df, df\ ErrorType2)$ ◁ apply clean_latitude_c3 that handle out of range violation

$df\ Others \leftarrow detect\ LatitudeViolation(df)$ ◁ general violation function for latitude

$df \leftarrow flag\_removal(df, df\ Others)$ ◁ flag values that still do not pass the violation detection for latitude

Attribution query

Example for attribution query result for merged data-cleaning workflow *WM*. A process id *Pid* with name *Process Name* created by *User Name* was derived from process id *Derived Pid*

```
% process attribution query
% for a workflow W, how many users contribute to the development
attribution(Rid,Pid,ProcessName,Pid,ProcessName,UserId,UserName) :-
recipe(Rid,_,Pid,_),
created_by(Pid,UserId),
user(UserId,UserName),
process(Pid,ProcessName).

attribution(Rid,Pid,ProcessName,PrevPid,PrevProcessName,UserId,UserName) :-
attribution(Rid,Pid,ProcessName,_,_,_,_),
tc_process(Pid,PrevPid),
user(UserId,UserName),
created_by(PrevPid,UserId),
process(PrevPid,PrevProcessName).

attribution_select(Rid,Pid,ProcessName,PrevPid,PrevProcessName,UserId,UserName):
-  attribution(Rid,Pid,ProcessName,PrevPid,PrevProcessName,UserId,UserName),
Rid={%w}.
```

Table 5: Attribution query result

Pid	Process Name	Derived Pid	Derived Process Name	User Name
p2	clean_neighbourhood_c2	p2	clean_neighbourhood_c2	C2
p9	clean_minimum_nights_c1	p9	clean_minimum_nights_c1	C1
p7	clean_number_of_reviews_c1	p7	clean_number_of_reviews_c1	C1
p16	clean_latitude_c1_c3	p16	clean_latitude_c1_c3	Integrator 1
p16	clean_latitude_c1_c3	p3	clean_latitude_c1	C1
p16	clean_latitude_c1_c3	p14	clean_latitude_c3	C3
p17	clean_longitude_c1_c3	p17	clean_longitude_c1_c3	Integrator 1
p17	clean_longitude_c1_c3	p5	clean_longitude_c1	C1
p17	clean_longitude_c1_c3	p15	clean_longitude_c3	C3
p18	clean_roomtype_c1_c3	p18	clean_roomtype_c1_c3	Integrator 1
p18	clean_roomtype_c1_c3	p11	clean_room_type_c1	C1
p18	clean_roomtype_c1_c3	p13	clean_room_type_c3	C3