CHARLES: Change-Aware Recovery of Latent Evolution Semantics in Relational Data

Shiyi He
University of Utah
Salt Lake City, Utah, USA
shiyi.he@utah.edu

Alexandra Meliou
University of Massachusetts Amherst
Amherst, Massachusetts, USA
ameli@cs.umass.edu

Anna Fariha
University of Utah
Salt Lake City, Utah, USA
afariha@cs.utah.edu

Abstract

Data-driven decision-making is at the core of many modern applications, and understanding the data is critical in supporting trust in these decisions. However, data is dynamic and evolving, just like the real-world entities it represents. Thus, an important component of understanding data is analyzing and drawing insights from the *changes* it undergoes. Existing methods for exploring data change list differences exhaustively, which are not interpretable by humans and lack salient insights regarding change trends.

We demonstrate CHARLES, a system that derives *semantic* summaries of changes between two snapshots of an evolving database, in an effective, concise, and interpretable way. Our key observation is that while datasets often evolve through point and other small-batch updates, rich data features can reveal *latent semantics* that can intuitively summarize the changes. Under the hood, CHARLES compares database versions, infers feasible transformations by fitting multiple regression lines over different data partitions to derive change summaries, and ranks them. CHARLES allows users to customize the ranking to obtain their preferred explanation by navigating the accuracy-interpretability tradeoff, and offers a proof of concept for reasoning about data evolution over real-world datasets.

Demo video: https://users.cs.utah.edu/~afariha/charles.mp4

name	gen	edu	exp	salary	bonus
Anne	F	PhD	2	$230,000	$23,000
Bob	M	PhD	3	$250,000	$25,000
Amber	F	MS	5	$160,000	$16,000
Allen	M	MS	1	$130,000	$13,000
Cathy	F	BS	2	$110,000	$11,000
Tom	M	MS	4	$150,000	$15,000
James	M	BS	3	$120,000	$12,000
Lucy	F	MS	4	$150,000	$15,000
Frank	M	PhD	1	$210,000	$21,000

(a) 2016 snapshot

name	gen	edu	exp	salary	bonus
Anne	F	PhD	3	$230,000	$25,150
Bob	M	PhD	4	$250,000	$27,250
Amber	F	MS	6	$160,000	$17,440
Allen	M	MS	2	$130,000	$13,790
Cathy	F	BS	3	$110,000	$11,000
Tom	M	MS	5	$150,000	$16,400
James	M	BS	4	$120,000	$12,000
Lucy	F	MS	5	$150,000	$16,400
Frank	M	PhD	2	$210,000	$23,050

(b) 2017 snapshot

Figure 1: Employee salaries have evolved over a year, with the *bonus* attribute increasing by 8–10% (highlighted in yellow). Context and trends of these changes are not apparent from the point updates.

1 Introduction

The task of data summarization is of prime importance to make sense of data. Existing data summarization systems enable users to interactively understand the content of a static database [4]. However, data is dynamic, evolving over time, and summarization techniques for static databases are ineffective at explaining this data evolution. Datasets often evolve through many tuple-level or small-batch-level updates. However, exhaustively listing all such fine-grained changes overwhelms human analysts. Fortunately, rich features in the data have the potential to concisely summarize such fine-grained changes, offering an "explanation" that captures the salient *semantics* of the data's evolution.

To understand change, we need to understand its cause (why), mechanism (how), and quantification (how much). This information is hard to extract from raw change logs, which are often unavailable to end users. Thus, understanding data changes via large change logs poses a significant hurdle for non-expert data consumers. Data versioning techniques can trace the locations and quantities of changes, but high-level trends are not typically obvious at that fine granularity. Instead, changes should be summarized at a coarser granularity to reveal the underlying causes and mechanisms.

EXAMPLE 1. *Figure 1 presents two snapshots of a salary database in (a) 2016 and (b) 2017. In 2016,* `bonus` *was a flat 10% of* `salary` *for all employees. In contrast, we observed no such straightforward trend in 2017. In some cases, the value of* `bonus` *differs from last year's value (highlighted in yellow), while in some cases they are identical (for Cathy and James). Furthermore, the difference ranges from 8% to 10% and is not identical for everyone. Simply knowing that* `bonus` *changed from last year leaves one unsatisfied, as it is not obvious what is the underlying trend behind such non-uniform changes. It turns out that the company adopted a new policy to reward long-serving employees and promote educational advancement based on three principles: (1) no one should receive lower bonus than the previous year, (2) employees with higher educational degree should be rewarded more, and (3) long-serving employees should be rewarded more.*

This is not immediately apparent by just looking at the data, since `bonus` *for 2017 is no longer directly tied to* `salary`*, as was the case in 2016. Instead, it is calculated based on a combination of last year's* `bonus`*, employee's education (*`edu`*), and years of experience (*`exp`*). Specifically, the following rules accurately explain the change trend:*

- ***R1**: Employees who have a PhD receive a 5% increase on last year's bonus, plus flat $1000.*
- ***R2**: Employees who have an MS and served for at least 3 years receive a 4% increase on last year's bonus, plus flat $800.*
- ***R3**: Employees who have an MS and served for less than 3 years receive a 3% increase on last year's bonus, plus flat $400.*

There are two desirable properties for a *change summary*: (1) it should be *precise*, i.e., be able to explain the changes *accurately*, and (2) it should be *interpretable* and *succinct* for easy human consumption. Note that there is a natural tension between these two desirable properties. Consider the following change summary:

- ***R4**: Everyone receives about 6% increase on last year's bonus.*

R4 is more interpretable (more succinct and human-consumable) than {**R1**, **R2**, **R3**}; however, **R4** does not accurately capture the change, while {**R1**, **R2**, **R3**} does. In contrast, one can provide a change summary by listing each individual cell that changed. However, such a summary—despite being very precise—would lack interpretability as this level of detail overwhelms the user.

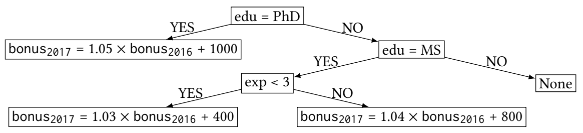

edu = PhD
YES
NO
bonus$_{2017}$ = 1.05 × bonus$_{2016}$ + 1000
edu = MS
YES
NO
None
exp < 3
YES
NO
bonus$_{2017}$ = 1.03 × bonus$_{2016}$ + 400
bonus$_{2017}$ = 1.04 × bonus$_{2016}$ + 800

Figure 2: A linear model tree explaining diff in datasets in Figure 1.

CHARLES. To meet the requirements of accuracy and interpretability, we developed CHARLES (Change-Aware Recovery of Latent Evolution Semantics), a system for producing a *semantic summary of changes* between two snapshots of a relational database, while striking a balance between accuracy and interpretability. Our key observation is that data changes are often driven by some underlying policies and the patterns within data evolution, as manifested by the changes, can potentially recover those policies. In this work, we focus on temporal changes and assume that given a *source* dataset (earlier version) and a *target* dataset (later version) of identical schema, the latter is obtained via a set of update operations over the former. Furthermore, we assume that no tuples are inserted or deleted[1]; only (numerical) values of various cells are altered[2].

The key challenge is to derive a partitioning of the tuples, such that tuples within each partition conform to a uniform "transformation" of reasonable complexity. As a proof of concept, CHARLES uses Linear Model Trees [8] to guide the search for data partitions based on a subset of data attributes (e.g., education and year of experience) and obtain a linear regression model at each leaf of the tree (e.g., bonus$_{2017}$ = 1.05 × bonus$_{2016}$ + 1000). The output is a tree (Figure 2), where the path from the root to a leaf defines a partition and the leaf defines the transformation (a linear model).

Furthermore, CHARLES enhances user experience by (1) *customization*—users can specify system parameters such as the maximum number of attributes they want to see in the change summary—and (2) *visualization*—they can interactively inspect different partitions of the data and the corresponding change trends.

Limitations. CHARLES focuses on finding an interpretable summary of data changes based only on the data, without any knowledge of external information. While the change summary produced by CHARLES may not always match the factual explanation (e.g., when change is due to some external factors), it nevertheless helps facilitate the development of hypotheses about the underlying causes of these changes. While CHARLES relies on linear models to capture change trends, this can be extended by augmenting the data with nonlinear features. However, nonlinear models are less interpretable, which justifies our choice of linear models.

Related work. Prior work [1] explores the history of changes in a database, but is limited to syntactic changes, suitable for historical change exploration involving a particular entity. Database comparator tools [7] and version control systems [3] only look for syntactic changes—changed values, altered objects, removal/addition of rows—which is not concise enough to provide high-level insights of the changes. Data-diff [11] explores change in data distributions in the context of data wrangling. But, these works cannot on semantically summarize changes between two database versions.

[1]This assumption often holds for a vast majority of tuples for many databases (e.g., databases of employees, products, countries, etc.). Even when it doesn't hold, we can focus change summarization involving the entities that are present in both versions.
[2]Non-numeric attributes may also change, however, this work focuses on explaining numerical changes as those are the hardest for humans to comprehend intuitively.

Explain-Da-V [10] is closest to CHARLES as it also explains transformations that convert a source dataset to a target dataset. However, it makes a strict assumption that changes are programmatically achieved by data scientists while achieving data-science tasks. As such, it focuses on the semantics of schema and data format transformations (e.g., data extraction, row deletion, adding attributes representing length of an attribute). We found Explain-Da-V to fail even for the toy dataset of Figure 1, as it does not support disjoint linear transformations such as {**R1**, **R2**, **R3**} of Example 1. In contrast, CHARLES can discover disjoint changes involving numeric attributes where the values of an attribute change based on interactions among other attributes, which represent natural evolution of real world entities. Furthermore, Explain-Da-V uses only accuracy as the primary objective and explainability just as a tie-breaker. In contrast, CHARLES allows the users to specify their desired balance between accuracy and explainability.

Local explanations for ML models [9] is inapplicable in our use case as they focus on classification. In contrast, we focus on the challenging problem of *pattern-based grouping*, where we need to find groups where elements share the same change patterns. However, a cyclic dependency exists here: change patterns can only be discovered once tuple groups are formed, but the appropriateness of the groups depends on the consistency of their change patterns.

Demonstration. In our demonstration, participants will witness how CHARLES generates change summaries from two datasets, tailored to user preferences, and enables them to effectively gain insights about data changes. We proceed to describe our solution sketch in Section 2 and then outline the demonstration scenario in Section 3.

2 Solution Sketch

Given a source dataset $\mathcal{D}_s$ and a target dataset $\mathcal{D}_t$ of identical schema, and a numerical attribute of interest a_i, we aim to produce a *ranked list* of *change summaries* that capture the changes observed between $\mathcal{D}_s(a_i)$ and $\mathcal{D}_t(a_i)$. Each change summary consists of a set of *transformations* over different data partitions. We rank the summaries based on their *scores*, which indicate how well they can strike a balance between accuracy and interpretability. As justified before, we assume that $\mathcal{D}_s$ and $\mathcal{D}_t$ contain the same real-world entities, i.e., only values of non-primary-key attributes were modified, and there were no insertions or deletions of tuples.

Change summary and conditional transformation. Our unit of explanation within a change summary is a *conditional transformation* (CT), which comprises a *condition* and a *transformation.* A summary $S = \{CT_1, CT_2, \dots\}$ comprises a set of CTs. The condition explains why a change happened, and the transformation describes the change itself. For instance, the following CT explains that employees with a PhD got 5% increase in bonus plus \$1000.

$$\underbrace{\text{edu = PhD}}_{\text{Condition}} \longrightarrow \underbrace{\text{new\_bonus = 1.05} \times \text{old\_bonus + 1000}}_{\text{Transformation}}$$

Desiderata for change summary. A desirable change summary must ensure that (1) all or most of the data changes are accurately covered and (2) the summary itself is interpretable and succinct for human consumption. To this end, we introduce $Score(S) \in [0, 1]$ for a summary S, which indicates how well S can represent the differences between $\mathcal{D}_s(a_i)$ and $\mathcal{D}_t(a_i)$.

$$Score(S) = \alpha \times Accuracy(S) + (1 - \alpha) \times Interpretability(S) \quad (1)$$

Here, α is a system parameter that controls the interplay between accuracy and interpretability.

Modeling accuracy. We model *Accuracy* by the inverse L_1 distance between $\hat{\mathcal{D}}_s(a_i)$ and $\mathcal{D}_t(a_i)$, where $\hat{\mathcal{D}}_s$ is the transformed dataset obtained by applying the CTs in S on $\mathcal{D}_s$. Other distance metrics can be plugged in here, we chose L_1 distance for simplicity.

Modeling interpretability. We prioritize the following characteristics of summaries:

- *Smaller summaries.* A summary with fewer CTs is preferable, as it leads to increased conciseness.
- *Simpler conditions and transformations.* A condition consists of a series of descriptors that identify specific segments of the data, so we prefer a simpler condition with fewer descriptors. E.g., the transformation "All Female employees received 5% bonus" is more interpretable than "All Asian, European Females, or Females working in HR received a 5% bonus". Similarly, transformations with fewer variables in the linear equation is preferred.
- *Conditions with higher data coverage.* A condition that yields a small data partition explains little of the change. Thus, we prefer conditions with higher coverage, yielding larger partitions.
- *Higher "normality" for conditions and transformations.* Conditions and transformations may involve numeric constants. We prefer the ones involving more "normal" values. E.g., the condition "Age > 25" is more normal than "Age > 23.796", and 5% for a salary increase is more normal (and interpretable) than 2.479%. We rely on domain expertise to model such notions of normality.

Then the interpretability score of a summary S is computed using the following equation[3], where $\beta_1, \ldots, \beta_4$ are tunable parameters:

$$\begin{aligned} Interpretability(S) = {} & \beta_1 \cdot Size(S) + \beta_2 \cdot Simplicity(S) \\ & + \beta_3 \cdot Coverage(S) + \beta_4 \cdot Normality(S) \end{aligned}$$

System parameters. Enhancing interpretability without significantly compromising accuracy results in a more effective summary, with a higher overall score. The parameter α controls the tradeoff between accuracy and interpretability, and the parameters $\beta_1, \ldots, \beta_4$ control the relative weights of various interpretability components. While the user is free to tune these parameters based on their preferences, we set default values of these parameters (e.g., we set α to 0.5 to assign equal weight to interpretability and accuracy) to cater to a diverse audience, allowing novices to bypass system parameter tuning and experts to adjust parameters.

CHARLES. CHARLES consists of two components: the *setup assistant*, which helps the users (optionally) tune the system parameters, and the *diff discovery engine*, which is responsible for generating the change summaries.

Setup assistant. For datasets with many attributes, the search space for possible summaries can be large. Users can help narrow down the set of relevant attributes, which has the dual benefits of reducing the search space, and pivoting the change summaries around attributes of interest. However, users who are unfamiliar with the schema may fail to do so. CHARLES addresses this challenge by estimating the influence of each attribute on the target attribute using correlation analysis and presents to the user a shortlist of most promising condition ($\mathcal{A}_{cond}$) and transformation ($\mathcal{A}_{tran}$) attributes for explaining the changes (Steps ④ & ⑤ in Figure 3).

CHARLES also allows the user to narrow down or expand the candidate attributes using two parameters: (1) c, the maximum number of attributes $\in \mathcal{A}_{cond}$ to use for partition discovery, and (2) t, the maximum number of numerical attributes $\in \mathcal{A}_{tran}$ to use to fit the linear model within each partition (Step ③). Additionally, CHARLES also sets the default values for other system parameters such as $\alpha, \beta_1, \ldots, \beta_4$ based on optimal empirical findings (Step ⑥).

Diff discovery engine. Instead of directly finding the best-scoring summaries via an optimization technique, CHARLES follows a two-step procedure: (1) generate a diverse set of candidate summaries, and (2) rank them based on their scores, and return the top-k.

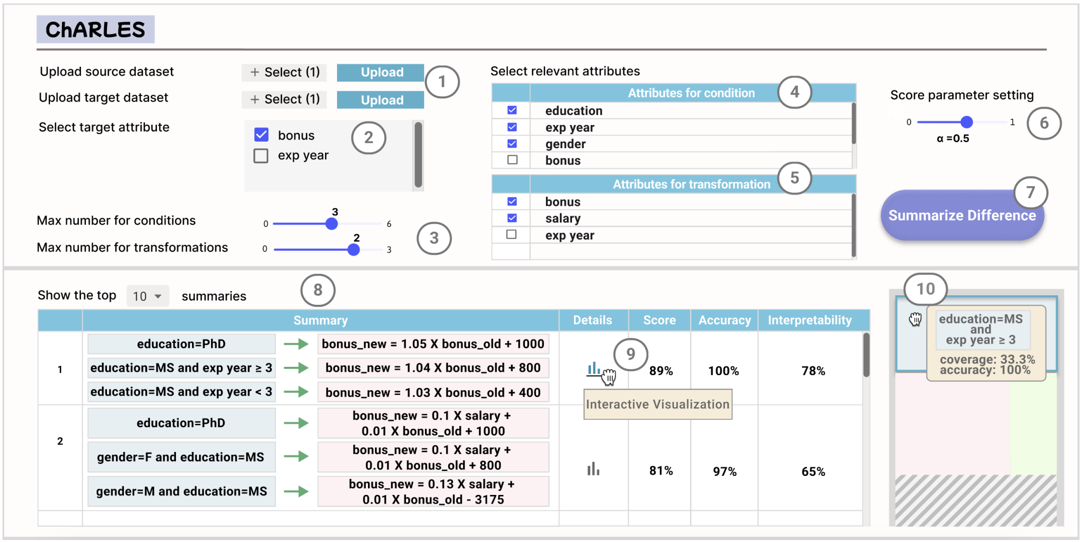
To this end, CHARLES first adopts Linear Model Tree (LMT) [8], which uses a greedy strategy to obtain a tree (Figure 2). Each leaf denotes a partition and a conjunction of the conditions along the path from the root to a leaf defines that leaf's partition's condition. At each leaf, a linear regression model is fitted based on the available transformation attributes A_{tran}. All such transformations over different partitions (leaves), together, form a change summary consisting of multiple conditional transformations.

Like decision trees, at each node, LMT *greedily* decides on the best partitioning scheme, over the available condition attributes A_{cond}, only based on the next-level accuracy (inverse of L_1 distance), which is computed by prematurely fitting a linear regression line at each child node and aggregating their regression errors. However, this can be sub-optimal because further splitting of a child node can drastically increase its accuracy, but is impossible to know ahead of time. Furthermore, since LMT only focuses on accuracy, it may produce summaries that are less interpretable.

To tackle this challenge, CHARLES uses the idea of ensembles and random forest [6]. Specifically, CHARLES guides construction of a sufficiently large, and *diverse* set of LMTs by randomizing the subsets of data attributes each LMT can use for partitioning and generating transformations. Based on $\mathcal{A}_{cond}$, $\mathcal{A}_{tran}$, and the parameters c and t , CHARLES enumerates all possible combinations of attributes to use for partitioning and generating transformations and utilizes a random subset of those combinations. E.g., for $c = 3$ and $t = 2$, CHARLES will consider all subsets of $\mathcal{A}_{cond}$ with cardinality ≤ 3 as partitioning attributes and all subsets of $\mathcal{A}_{tran}$ with cardinality ≤ 2 as transformation attributes. During the ensemble process, CHARLES limits possible options for each LMT in diverse ways. This produces a diverse set of LMTs, each representing a candidate change summary. CHARLES then scores these summaries using Equation 1, ranks them, and returns the top-k best summaries.

System efficiency. CHARLES greedily constructs the LMTs where the leaves of the LMTs use simple linear regression, which makes computation of each LMT very fast, with an amortized runtime complexity of $O(PNM^2)$, where P is the number of possible ways to split the data, N is the number of rows and M is the number of attributes in the data. Furthermore, the construction of random forest is embarrassingly parallelizable, causing minimal impact on the runtime performance.

[3]We omit the implementation details of $Size(.)$, $Simplicity(.)$, $Coverage(.)$, and $Normality(.)$ due to space limitations, however, these implementations are orthogonal to CHARLES and any user-defined implementations can be plugged in.


ChARLES

Upload source dataset + Select (1) Upload

Upload target dataset + Select (1) Upload

Select target attribute ☑ bonus ☐ exp year

Max number for conditions 3 (0–6)

Max number for transformations 2 (0–3)

Select relevant attributes

	Attributes for condition
☑	education
☑	exp year
☑	gender
☐	bonus

	Attributes for transformation
☑	bonus
☑	salary
☐	exp year

Score parameter setting α =0.5

Summarize Difference

Show the top 10 summaries

	Summary			Details	Score	Accuracy	Interpretability
1	education=PhD	→	bonus_new = 1.05 X bonus_old + 1000		89%	100%	78%
	education=MS and exp year ≥ 3	→	bonus_new = 1.04 X bonus_old + 800				
	education=MS and exp year < 3	→	bonus_new = 1.03 X bonus_old + 400				
2	education=PhD	→	bonus_new = 0.1 X salary + 0.01 X bonus_old + 1000		81%	97%	65%
	gender=F and education=MS	→	bonus_new = 0.1 X salary + 0.01 X bonus_old + 800				
	gender=M and education=MS	→	bonus_new = 0.13 X salary + 0.01 X bonus_old - 3175				

Figure 3: The CHARLES demo: (1) upload datasets, (2) select the target attribute, (3) specify the maximum number of attributes for condition and transformation, (4) CHARLES selects attributes for condition automatically, (5) CHARLES selects attributes for transformation automatically, (6) tune score parameter α, (7) request change summaries, (8) CHARLES presents a list of ranked summaries, with their overall scores, and scores for accuracy and interpretability, (9) click on a summary for more details, (10) detailed visualization of data partitions.

3 Demonstration

We will demonstrate CHARLES on a real-world dataset [5] representing salary information of employees of Montgomery County, MD for 2016 and 2017. The dataset contains 8 attributes: Department, Department Name, Division, Gender, Base Salary, Overtime Pay, Longevity Pay, and Grade. Figure 3 shows CHARLES's interface. During the demonstration, we will guide the participants through ten steps, each annotated with a circle in Figure 3.

Step (1) (Uploading datasets) The user uploads two dataset versions they want to compare. For ease of exposition, we use the toy datasets of Example 1 in this demo scenario.

Step (2) (Selecting the target attribute) Next, the user chooses the target attribute that manifests changes they wish to investigate. For our scenario, the user chooses "bonus".

Step (3) (Setting parameters) Next, the user chooses the maximum number of condition attributes to use for partitioning (3) and the maximum number of transformation attributes (2).

Steps (4) & (5) (Attribute selection) CHARLES presents a ranked list of attributes that are most promising for partitioning (4) and generating transformations (5). The user selects the top 3 from the first list and top 2 from the second, which is the default selection by CHARLES: "education", "exp year", and "gender" as potential condition attributes and "bonus" (of the previous year) and "salary" as potential transformation attributes.

Steps (6)–(8) (Change summaries) α represents the weight of accuracy in the *Score* function (set to 0.5 by default). Users can modify it if they wish (6). For example, for a more interpretable summary, they can set α to a lower value to shift the balance towards interpretability at the expense of accuracy. Upon user request (7), CHARLES displays the summaries (8), each comprising a set of conditional transformations—where conditions are in blue and transformations are in pink—followed by an option to visualize (9), overall score, accuracy, and interpretability scores. Here, the first summary reflects the scenario described in Example 1, which incurs a very high score of 89%. By default, CHARLES presents the 10 top-scoring summaries.

Steps (9) & (10) (Visualization) To better understand a summary, the user requests more details (9). CHARLES offers an interactive visualization (10) comprising several non-overlapping rectangles, each representing a data partition whose size corresponds to its data coverage. E.g., 33.3% employees fall within the top partition. For each partition, additional details—such as partitioning condition, data coverage, accuracy of the transformation—are revealed when the user hovers over the rectangle. The bottom partition, marked by diagonal patterns, indicates that no change was observed there.

Demonstration engagement. Our target users are data analysts, decision makers, and data enthusiasts who want to understand data change trends. After our guided demonstration, participants can plug their own datasets into CHARLES. We will also make additional datasets [2] available. Through the demonstration, we will showcase how CHARLES can semantically summarize data changes.

References

[1] T. Bleifuß, L. Bornemann, T. Johnson, D. Kalashnikov, F. Naumann, and D. Srivastava. 2018. Exploring Change - A New Dimension of Data Analytics. *PVLDB* (2018).
[2] Forbes World's Billionaires List [n. d.]. `https://www.forbes.com/billionaires/`.
[3] S. Huang, L. Xu, J. Liu, A. Elmore, and A. Parameswaran. 2017. OrpheusDB: Bolt-on Versioning for Relational Databases. *PVLDB* (2017).
[4] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. 2019. Interactive Data Exploration with Smart Drill-Down. *IEEE* (2019).
[5] Montgomery Police Dataset [n. d.]. `https://data.montgomerycountymd.gov/`.
[6] A Parmar, R Katariya, and V Patel. 2019. A review on random forest: An ensemble classifier. In *ICICI*. Springer, 758–763.
[7] PostgresCompare [n. d.]. `www.postgrescompare.com/`.
[8] D. Potts. 2004. Incremental learning of linear model trees. In *ICML*, Vol. 69. ACM.
[9] M. Ribeiro, S. Singh, and C. Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *SIGKDD*. 1135–1144.
[10] R. Shraga and R. Miller. 2023. Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V. *PVLDB* (2023).
[11] C. Sutton, T. Hobson, J. Geddes, and R. Caruana. 2018. Data Diff: Interpretable, Executable Summaries of Changes in Distributions for Data Wrangling. In *SIGKDD*.