Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems*

Technical Report

Anna Fariha[†‡]
University of Massachusetts
Amherst, MA, USA
afariha@cs.umass.edu

Ashish Tiwari[‡]
Arjun Radhakrishna
Sumit Gulwani
Microsoft
{astiwar,arradha,sumitg}@microsoft.com

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

ABSTRACT

The reliability of inferences made by data-driven systems hinges on the data's continued conformance to the systems' initial settings and assumptions. When serving data (on which we want to apply inference) deviates from the profile of the initial training data, the outcome of inference becomes unreliable. We introduce *conformance constraints*, a new data profiling primitive tailored towards quantifying the degree of *non-conformance*, which can effectively characterize if inference over that tuple is *untrustworthy*. Conformance constraints are constraints over certain arithmetic expressions (called *projections*) involving the numerical attributes of a dataset, which existing data profiling primitives such as functional dependencies and denial constraints cannot model.

The key finding we present is that projections that incur *low variance* on a dataset construct effective conformance constraints. This principle yields the surprising result that low-variance components of a principal component analysis, which are usually discarded for dimensionality reduction, generate stronger conformance constraints than the high-variance components. Based on this result, we provide a highly scalable and efficient technique—linear in data size and cubic in the number of attributes—for discovering conformance constraints for a dataset. To measure the degree of a tuple's non-conformance with respect to a dataset, we propose a *quantitative semantics* that captures how much a tuple violates the conformance constraints of that dataset. We demonstrate the value of conformance constraints on two applications: *trusted machine learning* and *data drift*. We empirically show that conformance constraints offer mechanisms to (1) reliably detect tuples on which the inference of a machine-learned model should not be trusted, and (2) quantify data drift more accurately than the state of the art.

1 INTRODUCTION

Data is central to modern systems in a wide range of domains, including healthcare, transportation, and finance. The core of modern data-driven systems typically comprises of models learned from large datasets, and they are usually optimized to target particular data and workloads. While these data-driven systems have seen wide adoption and success, their reliability and proper function hinge on the data's continued conformance to the systems' initial settings and assumptions. If the serving data (on which the system operates) deviates from the profile of the initial data (on which the system was trained), then system performance degrades and system behavior becomes unreliable. A mechanism to assess the trustworthiness of a system's inferences is paramount, especially for systems that perform safety-critical or high-impact operations.

A machine-learned (ML) model typically works best if the serving dataset follows the profile of the dataset the model was trained on; when it doesn't, the model's inference can be unreliable. One can profile a dataset in many ways, such as by modeling the data distribution of the dataset, or by finding the (implicit) *constraints* that the dataset satisfies. Distribution-oriented approaches learn data likelihood (e.g., joint or conditional distribution) from the training data, and can be used to check if the serving data is unlikely. An unlikely tuple does not necessarily imply that the model would fail for it. The problem with the distribution-oriented approaches is that they tend to overfit, and thus, are overly conservative towards unseen tuples, leading them to report many such false positives.

We argue that certain constraints offer a more effective and robust mechanism to quantify trust of a model's inference on a serving tuple. The reason is that learning systems implicitly exploit such constraints during model training, and build models that assume that the constraints will continue to hold for serving data. For example, when there exist high correlations among attributes in the training data, learning systems will likely reduce the weights assigned to redundant attributes that can be deduced from others, or eliminate them altogether through dimensionality reduction. If the serving data preserves the same correlations, such operations are inconsequential; otherwise, we may observe model failure.

In this paper, we characterize datasets with a new data-profiling primitive, *conformance constraints*, and we present a mechanism to identify *strong* conformance constraints, whose violation indicates unreliable inference. Conformance constraints specify constraints over *arithmetic relationships* involving multiple numerical attributes of a dataset. We argue that a tuple's conformance to the conformance constraints is more critical for accurate inference than its conformance to the training data distribution. This is because any violation of conformance constraints is likely to result in a catastrophic failure of a learned model that is built upon the assumption that the conformance constraints will always hold. Thus, we can use a tuple's deviation from the conformance constraints as a proxy for the trust on a learned model's inference for that tuple. We proceed

*An earlier version of this paper had a different title: "Data Invariants: On Trust in Data-Driven Systems".
†Work done while the author was an intern at Microsoft.
‡Both authors contributed equally to this research.

	Departure Date	Departure Time [DT]	Arrival Time [AT]	Duration (min) [DUR]
t_1	May 2	14:30	18:20	230
t_2	July 22	09:05	12:15	195
t_3	June 6	10:20	12:20	115
t_4	May 19	11:10	13:05	117
t_5	April 7	22:30	06:10	458

Figure 1: Sample of the airlines dataset (details are in Section 6.1), showing departure, arrival, and duration only. The dataset does not report arrival date, but an arrival time earlier than departure time (e.g., last row), indicates an overnight flight. All times are in 24 hour format and in the same time zone. There is some noise in the values.

to describe a real-world example of conformance constraints, drawn from our case-study evaluation on *trusted machine learning* (TML).

Example 1. *We used a dataset with flight information that includes data on departure and arrival times, flight duration, etc. (Fig. 1) to train a linear regression model to predict flight delays. The model was trained on a subset of the data that happened to include only daytime flights (such as the first four tuples). In an empirical evaluation of the regression accuracy, we found that the mean absolute error of the regression output more than quadruples for overnight flights (such as the last tuple t_5), compared to daytime flights. The reason is that tuples representing overnight flights deviate from the profile of the training data that only contained daytime flights. Specifically, daytime flights satisfy the conformance constraint that "arrival time is later than departure time and their difference is very close to the flight duration", which does not hold for overnight flights. Note that this constraint is just based on the covariates (predictors) and does not involve the target attribute delay. Critically, although this conformance constraint is unaware of the regression task, it was still a good proxy of the regressor's performance. In contrast, approaches that model data likelihood may report long daytime flights as unlikely, since all flights in the training data (t_1–t_4) were also short flights, resulting in false alarms, as the model works very well for most daytime flights, regardless of the duration (i.e., for both short and long daytime flights).*

Example 1 demonstrates that when training data has *coincidental* relationships (e.g., the one between AT, DT, and DUR for daytime flights), then ML models may *implicitly* assume them as *invariants*. Conformance constraints can capture such data invariants and flag non-conforming tuples (overnight flights) during serving.

Conformance constraints. Conformance constraints complement the existing data profiling literature, as the existing constraint models, such as functional dependencies and denial constraints, cannot model arithmetic relationships. For example, the conformance constraint of Example 1 is: $-\epsilon_1 \leq AT - DT - DUR \leq \epsilon_2$, where ϵ_1 and ϵ_2 are small values. Conformance constraints can capture complex linear dependencies across attributes within a *noisy* dataset. For example, if the flight departure and arrival data reported the hours and the minutes across separate attributes, the constraint would be on a different arithmetic expression: $(60 \cdot arrHour + arrMin) - (60 \cdot depHour + depMin) - duration$.

The core component of a conformance constraint is the arithmetic expression, called *projection*, which is obtained by a linear combination of the numerical attributes. There is an unbounded number of projections that we can use to form arbitrary conformance constraints. For example, for the projection AT, we can find a broad range $[\epsilon_3, \epsilon_4]$, such that all training tuples in Example 1 satisfy the conformance constraint $\epsilon_3 \leq AT \leq \epsilon_4$. However, this constraint is too inclusive and a learned model is unlikely to exploit such a weak constraint. In contrast, the projection $AT - DT - DUR$ leads to a stronger conformance constraint with a narrow range as its bounds, which is selectively permissible, and thus, more effective.

Challenges and solution sketch. The principal challenge is to discover an *effective* set of conformance constraints that are likely to affect a model's inference implicitly. We first characterize "good" projections (that construct effective constraints) and then propose a method to discover them. We establish through theoretical analysis two important results: (1) A projection is good over a dataset if it is almost constant (i.e., has low variance) for all tuples in that dataset. (2) A set of projections, collectively, is good if the projections have small pair-wise correlations. We show that low variance components of a principal component analysis (PCA) on a dataset yield such a set of projections. Note that this is different from—and in fact completely opposite to—the traditional approaches (e.g., [63]) that perform multidimensional analysis based on the high-variance principal components, after reducing dimensionality using PCA.

Scope. Fig. 2 summarizes prior work on related problems, but the scope of our setting differs significantly. Specifically, we can detect if a serving tuple is non-conforming with respect to the training dataset *only based on its predictor attributes*, and require no knowledge of the ground truth. This setting is essential in many practical applications when we observe *extreme verification latency* [74], where ground truths for serving tuples are not immediately available. For example, consider a self-driving car that is using a trained controller to generate actions based on readings of velocity, relative positions of obstacles, and their velocities. In this case, we need to determine, only based on the sensor readings (predictors), when the driver should be alerted to take over vehicle control, as we cannot use ground-truths to generate an alert.

Furthermore, we *do not assume access to the model*, i.e., model's predictions on a given tuple. This setting is necessary for (1) safety-critical applications, where the goal is to quickly alert the user, without waiting for the availability of the prediction, (2) auditing and privacy-preserving applications where the prediction cannot be shared, and (3) when we are unaware of the detailed functionality of the system due to privacy concerns or lack of jurisdiction, but only have some meta-information such as the system trains some linear model over the training data.

We focus on identifying *tuple-level* non-conformance as opposed to dataset-level non-conformance that usually requires observing entire data's distribution. However, our tuple-level approach trivially extends (by aggregation) to the entire dataset.

Contrast with prior art. We now discuss where conformance constraints fit with respect to the existing literature (Fig. 2) on data profiling and literature on modeling trust in data-driven inferences

Data profiling techniques. Conformance constraints fall under the umbrella of data profiling, which refers to the task of extracting technical metadata about a given dataset [5]. A key task in data profiling is to learn relationships among attributes. Functional dependencies (FD) [59] and their variants only capture if a relationship exists between two sets of attributes, but do not provide a

Legend		constraints						violation		setting			technique					TML	
HP: Hyper Parameter FD: Functional Dependency DC: Denial Constraint ∅: Does not require ⊥: Not applicable ★: Supports via extension !: Partially		parametric	arithmetic	approximate	conditional	notion of weight	interpretable	continuous	tuple-wise	noisy data	numerical attr.	categorial attr.	∅thresholds	∅distance metric	∅HP tuning	scalable	task agnostic	∅access to model
Data Profiling	**Conformance Constraints**	✓	✓	✓	✓	✓	★	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	FD [59]						✓					✓	✓	✓	✓			
	Approximate FD [50]			✓			✓			✓		✓		✓	✓			
	Metric FD [48]			✓			✓			✓	✓	✓			⊥	⊥		
	Conditional FD [23]	!			✓		✓		✓	✓		✓		✓	✓			
	Pattern FD [62]	!					✓		✓	✓		✓		✓	✓			
	Soft FD [38]			✓		✓	✓	✓		✓	✓	✓		✓	✓	✓		
	Relaxed FD [16]			✓			✓			✓	✓	✓			✓			
	FDX [93]						✓			✓		✓				✓		
	Differential Dependency [72]						✓				✓	✓			✓			
	DC [13, 17]	!		✓	✓	✓	✓		✓	✓	✓	✓		✓	✓			
	Approximate DC [53, 61]	!		✓	✓	✓	✓		✓	✓	✓	✓		✓	✓			
	Statistical Constraint [91]			✓			✓	✓		✓	✓	✓		✓	✓	✓		
Learning	Ordinary Least Square	✓	✓	✓			★	✓	✓	✓	✓	★	✓	✓	✓	✓		
	Total Least Square	✓	✓	✓			★	✓	✓	✓	✓	★	✓	✓	✓	✓	✓	✓
	Auto-encoder [20]	⊥						✓	✓	✓	✓	✓				✓	✓	✓
	Schelter et al. [68]+	⊥						✓			✓	✓		✓	✓	✓		
	Jiang et al. [41]	⊥						✓	✓	✓	✓	✓	✓			✓		
	Hendrycks et al. [31]	⊥							✓	✓	✓	✓	✓	✓	✓	✓		
	Model's Prediction Probability	⊥						✓	✓	varies								

not addressed in prior work

+ Requires additional information

Figure 2: Conformance constraints complement existing data profiling primitives and provide an efficient mechanism to quantify trust in prediction, with minimal assumption on the setting.

closed-form (parametric) expression of the relationship. Using the FD $\{AT, DT\} \rightarrow \{DUR\}$ to model the constraint of Example 1 suffers from several limitations. First, since the data is noisy, no exact FD can be learned. Metric FDs [48] allow small variations in the data (similar attribute values are considered identical), but hinge on appropriate distance metrics and thresholds. For example, if *time* is split across two attributes (*hour* and *minute*), the distance metric is non-trivial: it needs to encode that $\langle hour = 4, min = 59\rangle$ and $\langle hour = 5, min = 1\rangle$ are similar, while $\langle hour = 4, min = 1\rangle$ and $\langle hour = 5, min = 59\rangle$ are not. In contrast, conformance constraints can model the composite attribute $(60 \cdot hour + minute)$ by automatically discovering the coefficients 60 and 1 for such a composite attribute.

Denial constraints (DC) [13, 17, 53, 61] encapsulate a number of different data-profiling primitives such as FDs and their variants (e.g, [23]). Exact DCs can adjust to noisy data by adding predicates until the constraint becomes exact over the entire dataset, but this can make the constraint extremely large and complex, which might even fail to provide the desired generalization. For example, a finite DC—whose language is limited to universally-quantified first-order logic—cannot model the constraint of Example 1, which involves an arithmetic expression (addition and multiplication with a constant). Expressing conformance constraints requires a richer language that includes linear arithmetic expressions. Pattern functional dependencies (PFD) [62] move towards addressing this limitation of DCs, but they focus on text attributes: they are regex-based and treat digits as characters. However, modeling arithmetic relationships of numerical attributes requires interpreting digits as numbers.

To adjust for noise, FDs and DCs either relax the notion of constraint violation or allow a user-defined fraction of tuples to violate the (strict) constraint [16, 36, 38, 48, 50, 53, 61]. Some approaches [38, 91, 93] use statistical techniques to model other types of data profiles such as correlations and conditional dependencies. However, they require additional parameters such as noise and violation thresholds and distance metrics. In contrast, conformance constraints do not require any parameter from the user and work on noisy datasets.

Existing data profiling techniques are not designed to learn what ML models exploit and they are sensitive to noise in the numerical attributes. Moreover, data constraint discovery algorithms typically search over an exponential set of candidates, and hence, are not scalable: their complexity grows exponentially with the number of attributes or quadratically with data size. In contrast, our technique for deriving conformance constraints is highly scalable (linear in data size) and efficient (cubic in the number of attributes). It does not explicitly explore the candidate space, as PCA—which lies at the core of our technique—performs the search *implicitly* by iteratively refining weaker constraints to stronger ones.

Learning techniques. While *ordinary least square* finds the lowest-variance projection, it minimizes observational error on only the target attribute, and thus, does not apply to our setting. *Total least square* offers a partial solution to our problem as it takes observational errors on all predictor attributes into account. However, it finds only one projection—the lowest variance one—that fits the data tuples best. But there may exist other projections with slightly higher variances and we consider them all. As we show empirically in Section 6.2, constraints derived from multiple projections, collectively, capture various aspects of the data, and result in an effective data profile targeted towards certain tasks such as data-drift quantification. (More discussion is in the Appendix.)

Contributions. We make the following contributions:

- We ground the motivation of our work with two case studies on trusted machine learning (TML) and data drift. (Section 2)
- We introduce and formalize conformance constraints, a new data profiling primitive that specify constraints over arithmetic relationships among numerical attributes of a dataset. We describe a *conformance language* to express conformance constraints, and a *quantitative semantics* to quantify how much a tuple violates the conformance constraints. In applications of constraint violations, some violations may be more or less critical than others. To capture that, we consider a notion of constraint importance, and weigh violations against constraints accordingly. (Section 3)
- We formally establish that strong conformance constraints are constructed from projections with small variance and small mutual correlation on the given dataset. Beyond simple linear constraints (e.g., the one in Example 1), we derive *disjunctive* constraints, which are disjunctions of linear constraints. We achieve this by dividing the dataset into disjoint partitions, and learning linear constraints for each partition. We provide an efficient, scalable, and highly parallelizable algorithm for computing a set of linear conformance constraints and disjunctions over them. We also analyze its runtime and memory complexity. (Section 4)
- We formalize the notion of *unsafe* tuples in the context of trusted machine learning and provide a mechanism to detect unsafe tuples using conformance constraints. (Section 5)
- We empirically analyze the effectiveness of conformance constraints in our two case-study applications—TML and data-drift quantification. We show that conformance constraints can reliably predict the trustworthiness of linear models and quantify data drift precisely, outperforming the state of the art. (Section 6)

2 CASE STUDIES

Like other data-profiling primitives, conformance constraints have general applicability in understanding and describing datasets. But their true power lies in quantifying the degree of a tuple's non-conformance with respect to a reference dataset. Within the scope of this paper, we focus on two case studies in particular to motivate our work: trusted machine learning and data drift. We provide an extensive evaluation over these applications in Section 6.

Trusted machine learning (TML) refers to the problem of quantifying trust in the inference made by a machine-learned model on a new serving tuple [41, 64, 67, 80, 86]. This is particularly useful in case of extreme verification latency [74], where ground-truth outputs for new serving tuples are not immediately available to evaluate the performance of a learned model, when auditing models for trustworthiness, and in privacy-preserving applications where even the model's predictions cannot be shared. When a model is trained using a dataset, the conformance constraints for that dataset specify a safety envelope [80] that characterizes the tuples for which the model is expected to make trustworthy predictions. If a serving tuple falls outside the safety envelope (violates the conformance constraints), then the model is likely to produce an untrustworthy inference. Intuitively, the higher the violation, the lower the trust. Some classifiers produce a confidence measure along with the class prediction, typically by applying a softmax function to the raw numeric prediction values. However, such a confidence measure is not well-calibrated [28, 41], and therefore, cannot be reliably used as a measure of trust in the prediction. Additionally, a similar mechanism is not available for inferences made by regression models.

In the context of TML, we formalize the notion of *unsafe tuples*, on which the prediction may be untrustworthy. We establish that conformance constraints provide a sound and complete procedure for detecting unsafe tuples, which indicates that the search for conformance constraints should be guided by the class of models considered by the corresponding learning system (Section 5).

Data drift [10, 27, 51, 63] specifies a significant change in a dataset with respect to a reference dataset, which typically requires that systems be updated and models retrained. Aggregating tuple-level non-conformances over a dataset gives us a *dataset-level* non-conformance, which is an effective measurement of data drift. To quantify how much a dataset D' drifted from a reference dataset D, our three-step approach is: (1) compute conformance constraints for D, (2) evaluate the constraints on all tuples in D' and compute their violations (degrees of non-conformance), and (3) finally, aggregate the tuple-level violations to get a dataset-level violation. If all tuples in D' satisfy the constraints, then we have no evidence of drift. Otherwise, the aggregated violation serves as the drift quantity.

While we focus on these two applications here, we mention other applications of conformance constraints in the Appendix.

3 CONFORMANCE CONSTRAINTS

In this section, we define conformance constraints that allow us to capture complex arithmetic dependencies involving numerical attributes of a dataset. Then we propose a language for representing them. Finally, we define quantitative semantics over conformance constraints, which allows us to quantify their violation.

Basic notations. We use $\mathcal{R}(A_1, A_2, \ldots, A_m)$ to denote a relation schema where A_i denotes the i^{th} attribute of $\mathcal{R}$. We use $\texttt{Dom}_i$ to denote the domain of attribute A_i. Then the set $\mathbf{Dom}^m = \texttt{Dom}_1 \times \cdots \times \texttt{Dom}_m$ specifies the domain of all possible tuples. We use $t \in \mathbf{Dom}^m$ to denote a tuple in the schema $\mathcal{R}$. A dataset $D \subseteq \mathbf{Dom}^m$ is a specific instance of the schema $\mathcal{R}$. For ease of notation, we assume some order of tuples in D and we use $t_i \in D$ to refer to the i^{th} tuple and $t_i.A_j \in \texttt{Dom}_j$ to denote the value of the j^{th} attribute of t_i.

Conformance constraint. A conformance constraint Φ characterizes a set of allowable or conforming tuples and is expressed through a *conformance language* (Section 3.1). We write $\Phi(t)$ and $\neg\Phi(t)$ to denote that t satisfies and violates Φ, respectively.

Definition 2 (Conformance constraint). *A conformance constraint for a dataset $D \subseteq \mathbf{Dom}^m$ is a formula $\Phi : \mathbf{Dom}^m \mapsto \{\texttt{True}, \texttt{False}\}$ such that $|\{t \in D \mid \neg\Phi(t)\}| \ll |D|$.*

The set $\{t \in D \mid \neg\Phi(t)\}$ denotes atypical tuples in D that do not satisfy the conformance constraint Φ. In our work, we do not need to know the set of atypical tuples, nor do we need to purge the atypical tuples from the dataset. Our techniques derive constraints in ways that ensure there are very few atypical tuples (Section 4).

3.1 Conformance Language

Projection. A central concept in our conformance language is *projection*. Intuitively, a projection is a derived attribute that specifies a "lens" through which we look at the tuples. More formally, a projection is a function $F : \mathbf{Dom}^m \mapsto \mathbb{R}$ that maps a tuple $t \in \mathbf{Dom}^m$ to a real number $F(t) \in \mathbb{R}$. In our language for conformance constraints, we only consider projections that correspond to linear combinations of the numerical attributes of a dataset. Specifically, to define a projection, we need a set of numerical coefficients for all attributes of the dataset and the projection is defined as a sum over the attributes, weighted by their corresponding coefficients. We extend a projection F to a dataset D by defining $F(D)$ to be the sequence of reals obtained by applying F on each tuple in D individually.

Grammar. Our language for conformance constraints consists of formulas Φ generated by the following grammar:

$$\begin{array}{rcl} \phi & := & \texttt{lb} \leq F(\vec{A}) \leq \texttt{ub} \mid \wedge(\phi, \ldots, \phi) \\ \psi_A & := & \vee(\ (A = c_1) \rhd \phi,\ (A = c_2) \rhd \phi,\ \ldots) \\ \Psi & := & \psi_A \mid \wedge(\psi_{A_1}, \psi_{A_2}, \ldots) \\ \Phi & := & \phi \mid \Psi \end{array}$$

The language consists of (1) bounded constraints $\texttt{lb} \leq F(\vec{A}) \leq \texttt{ub}$ where F is a projection on $\mathbf{Dom}^m$, $\vec{A}$ is the tuple of formal parameters $(A_1, A_2, \ldots, A_m)$, and $\texttt{lb}, \texttt{ub} \in \mathbb{R}$ are reals; (2) equality constraints $A = c$ where A is an attribute and c is a constant in A's domain; and (3) operators ($\rhd$, $\wedge$, and $\vee$,) that connect the constraints. Intuitively, $\rhd$ is a switch operator that specifies which constraint ϕ applies based on the value of the attribute A, $\wedge$ denotes conjunction, and $\vee$ denotes disjunction. Formulas generated by ϕ and Ψ are called *simple constraints* and *compound constraints*, respectively. Note that a formula generated by ψ_A only allows equality constraints on a single attribute, namely A, among all the disjuncts.

Example 3. *Consider the dataset D consisting of the first four tuples $\{t_1, t_2, t_3, t_4\}$ of Fig. 1. A simple constraint for D is:*

$$\phi_1 : -5 \leq AT - DT - DUR \leq 5.$$

Here, the projection $F(\vec{A}) = AT-DT-DUR$, with attribute coefficients $\langle 1, -1, -1\rangle$, $\texttt{lb} = -5$, and $\texttt{ub} = 5$. A compound constraint is:

$$\begin{aligned}\psi_2 : M = \text{“May”} \rhd -2 \leq\ & AT - DT - DUR \ \leq 0\\ \vee\ M = \text{“June”} \rhd\ \ 0 \leq\ & AT - DT - DUR \ \leq 5\\ \vee\ M = \text{“July”} \rhd -5 \leq\ & AT - DT - DUR \ \leq 0\end{aligned}$$

For ease of exposition, we assume that all times are converted to minutes (e.g., $\texttt{06:10} = 6\times60+10 = 370$) and M denotes the departure month, extracted from Departure Date.

Note that arithmetic expressions that specify linear combination of numerical attributes (highlighted above) are disallowed in denial constraints [17] which only allow raw attributes and constants (more details are in the Appendix).

3.2 Quantitative Semantics

Conformance constraints have a natural Boolean semantics: a tuple either satisfies a constraint or it does not. However, Boolean semantics is of limited use in practice, because it does not quantify the degree of constraint violation. We interpret conformance constraints using a quantitative semantics, which quantifies violations, and reacts to noise more gracefully than Boolean semantics.

The quantitative semantics $[\![\Phi]\!](t)$ is a measure of the violation of Φ on a tuple t—with a value of 0 indicating no violation and a value greater than 0 indicating some violation. In Boolean semantics, if $\Phi(t)$ is `True`, then $[\![\Phi]\!](t)$ will be 0; and if $\Phi(t)$ is `False`, then $[\![\Phi]\!](t)$ will be 1. Formally, $[\![\Phi]\!]$ is a mapping from $\mathbf{Dom}^m$ to $[0, 1]$.

Quantitative semantics of simple constraints. We build upon ϵ-insensitive loss [85] to define the quantitative semantics of simple constraints, where the bounds lb and ub define the ϵ-insensitive zone:[1]

$$[\![\texttt{lb} \leq F(\vec{A}) \leq \texttt{ub}]\!](t) := \eta(\alpha \cdot \max(0, F(t) - \texttt{ub}, \texttt{lb} - F(t)))$$

$$[\![\wedge(\phi_1, \ldots, \phi_K)]\!](t) := \textstyle\sum_k^K \gamma_k \cdot [\![\phi_k]\!](t)$$

Below, we describe the parameters of the quantitative semantics, and provide further details on them in the Appendix.

Scaling factor $\alpha \in \mathbb{R}^+$.
Projections are unconstrained functions and different projections can map the same tuple to vastly different values. We use a scaling factor α to standardize the values computed by a projection F, and to bring the values of different projections to the same comparable scale. The scaling factor is automatically computed as the inverse of the standard deviation: $\frac{1}{\sigma(F(D))}$. We set α to a large positive number when $\sigma(F(D)) = 0$.

Normalization function $\eta(.) : \mathbb{R} \mapsto [0, 1]$.
The normalization function maps values in the range $[0, \infty)$ to the range $[0, 1)$. While any monotone mapping from $\mathbb{R}^{\geq 0}$ to $[0, 1)$ can be used, we pick $\eta(z) = 1 - e^{-z}$.

Importance factors $\gamma_k \in \mathbb{R}^+$, $\sum_k^K \gamma_k$=1.
The weights γ_k control the contribution of each bounded-projection constraint in a conjunctive formula. This allows for prioritizing constraints that are more significant than others within the context of a particular application. In our work, we derive the importance factor of a constraint automatically, based on its projection's standard deviation over D.

[1] For a target value y, predicted value $\hat{y}$, and a parameter ϵ, the ϵ-insensitive loss is 0 if $|y - \hat{y}| < \epsilon$ and $|y - \hat{y}| - \epsilon$ otherwise.

Quantitative semantics of compound constraints. Compound constraints are first simplified into simple constraints, and they get their meaning from the simplified form. We define a function $\texttt{simp}(\psi, t)$ that takes a compound constraint ψ and a tuple t and returns a simple constraint. It is defined recursively as follows:

$$\texttt{simp}(\vee(\ (A = c_1) \rhd \phi_1, (A = c_2) \rhd \phi_2, \ldots), t) := \phi_k \text{ if } t.A = c_k$$

$$\texttt{simp}(\wedge(\psi_{A_1}, \psi_{A_2}, \ldots), t) := \wedge(\texttt{simp}(\psi_{A_1}, t), \texttt{simp}(\psi_{A_2}, t), \ldots)$$

If the condition in the definition above does not hold for any c_k, then $\texttt{simp}(\psi, t)$ is undefined and $\texttt{simp}(\wedge(\ldots, \psi, \ldots), t)$ is also undefined. If $\texttt{simp}(\psi, t)$ is undefined, then $[\![\psi]\!](t) := 1$. When $\texttt{simp}(\psi, t)$ is defined, the quantitative semantics of ψ is given by:

$$[\![\psi]\!](t) := [\![\texttt{simp}(\psi, t)]\!](t)$$

Since compound constraints simplify to simple constraints, we mostly focus on simple constraints. Even there, we pay special attention to bounded-projection constraints (ϕ) of the form $\texttt{lb} \leq F(\vec{A}) \leq \texttt{ub}$, which lie at the core of simple constraints.

Example 4. *Consider the constraint ϕ_1 from Example 3. For $t \in D$, $[\![\phi_1]\!](t) = 0$ since ϕ_1 is satisfied by all tuples in D. The standard deviation of the projection F over D, $\sigma(F(D))$=$\sigma(\{0, -5, 5, -2\})$=3.6. Now consider the last tuple $t_5 \notin D$. $F(t_5) = (370-1350)-458 = -1438$, which is way below the lower bound -5 of ϕ_1. Now we compute how much t_5 violates ϕ_1: $[\![\phi_1]\!](t_5) = [\![-5 \leq F(\vec{A}) \leq 5]\!](t_5) = \eta(\alpha \cdot \max(0, -1438 - 5, -5 + 1438)) = 1 - e^{-\frac{1433}{3.6}} \approx 1$. Intuitively, this implies that t_5 strongly violates ϕ_1.*

4 CONFORMANCE CONSTRAINT SYNTHESIS

In this section, we describe our techniques for deriving conformance constraints. We start with the synthesis of simple constraints (the ϕ constraints in our language specification), followed by compound constraints (the Ψ constraints in our language specification). Finally, we analyze the time and memory complexity of our algorithm.

4.1 Simple Conformance Constraints

Synthesizing simple conformance constraints involves (a) discovering the projections, and (b) discovering the lower and upper bounds for each projection. We start by discussing (b), followed by the principle to identify effective projections, based on which we solve (a).

4.1.1 Synthesizing Bounds for Projections. Fix a projection F and consider the bounded-projection constraint ϕ: $\texttt{lb} \leq F(\vec{A}) \leq \texttt{ub}$. Given a dataset D, a trivial choice for the bounds that are valid on all tuples in D is: $\texttt{lb} = \min(F(D))$ and $\texttt{ub} = \max(F(D))$. However, this choice is very sensitive to noise: adding a single atypical tuple to D can produce very different constraints. Instead, we use a more robust choice as follows:

$$\texttt{lb} = \mu(F(D)) - C \cdot \sigma(F(D)), \quad \texttt{ub} = \mu(F(D)) + C \cdot \sigma(F(D))$$

Here, $\mu(F(D))$ and $\sigma(F(D))$ denote the mean and standard deviation of the values in $F(D)$, respectively, and C is some positive constant. With these bounds, $[\![\phi]\!](t) = 0$ implies that $F(t)$ is within $C \times \sigma(F(D))$ from the mean $\mu(F(D))$. In our experiments, we set $C = 4$, which ensures that in expectation, very few tuples in D will violate the constraint for many distributions of the values in $F(D)$. Specifically, if $F(D)$ follows a normal distribution, then 99.99%

of the population is expected to lie within 4 standard deviations from mean. Note that we make no assumption on the original data distribution of each attribute.

Setting the bounds lb and ub as $C \cdot \sigma(F(D))$-away from the mean, and the scaling factor α as $\frac{1}{\sigma(F(D))}$, guarantees the following property for our quantitative semantics:

Lemma 5. *Let D be a dataset and let ϕ_k be $\mathtt{lb}_k \leq F_k(\vec{A}) \leq \mathtt{ub}_k$ for $k = 1, 2$. Then, for any tuple t, if $\frac{|F_1(t)-\mu(F_1(D))|}{\sigma(F_1(D))} \geq \frac{|F_2(t)-\mu(F_2(D))|}{\sigma(F_2(D))}$, then $[\![\phi_1]\!](t) \geq [\![\phi_2]\!](t)$.*

This means that larger deviation from the mean (proportionally to the standard deviation) results in higher degree of violation under our semantics. The proof follows from the fact that the normalization function $\eta(.)$ is monotonically increasing, and hence, $[\![\phi_k]\!](t)$ is a monotonically non-decreasing function of $\frac{|F_k(t)-\mu(F_k(D))|}{\sigma(F_k(D))}$.

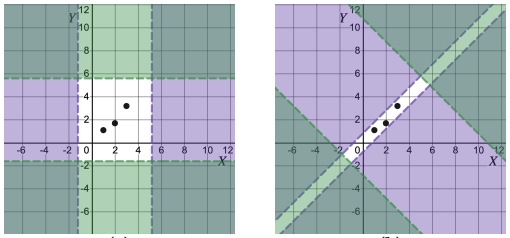
4.1.2 Principle for Synthesizing Projections. We start by investigating what makes a constraint more effective than others. An effective constraint (1) should not overfit the data, but rather generalize by capturing the properties of the data, and (2) should not underfit the data, because it would be too permissive and fail to identify deviations effectively. Our flexible bounds (Section 4.1.1) serve to avoid overfitting. In this section, we focus on identifying the principles that help us avoid underfitting. We first describe the key technical ideas for characterizing effective projections through example and then proceed to formalization.

Example 6. *Let D be a dataset of three tuples {(1,1.1),(2,1.7),(3,3.2)} with two attributes X and Y. Consider two arbitrary projections: X and Y. For X: $\mu(X(D)) = 2$ and $\sigma(X(D)) = 0.8$. So, bounds for its conformance constraint are:* $\mathtt{lb} = 2-4\times0.8 = -1.2$ *and* $\mathtt{ub} = 2+4\times0.8 = 5.2$. *This gives us the conformance constraint: $-1.2 \leq X \leq 5.2$. Similarly, for Y, we get the conformance constraint: $-1.6 \leq Y \leq 5.6$. Fig. 3(a) shows the conformance zone (clear region) defined by these two conformance constraints. The shaded region depicts non-conformance zone. The conformance zone is large and too permissive: it allows many atypical tuples with respect to D, such as $(0, 4)$ and $(4, 0)$.*

A natural question arises: are there other projections that can better characterize conformance with respect to the tuples in D? The answer is yes and next we show another pair of projections that shrink the conformance zone significantly.

Example 7. *In Fig. 3(b), the clear region is defined by the conformance constraints $-0.8 \leq X-Y \leq 0.8$ and $-2.8 \leq X+Y \leq 10.8$, over projections $X - Y$ and $X + Y$, respectively. The region is indeed much smaller than the one in Fig. 3(a) and allows fewer atypical tuples.*

How can we derive projection $X - Y$ from the projections X and Y, given D? Note that X and Y are highly correlated in D. In Lemma 11, we show that two highly-correlated projections can be linearly combined to construct another projection with lower standard deviation that generates a *stronger* constraint. We proceed to formalize stronger constraint—which defines whether a constraint is more effective than another in quantifying violation—and *incongruous* tuples—which help us estimate the subset of the data domain for which a constraint is stronger than the others.



(a) (b)

Figure 3: Clear and shaded regions depict conformance and non-conformance zones, respectively. (a) Correlated projections X and Y yield conformance constraints forming a large conformance zone, (b) Uncorrelated (orthogonal) projections $X - Y$ and $X + Y$ yield conformance constraints forming a smaller conformance zone.

Definition 8 (Stronger constraint). *A conformance constraint ϕ_1 is stronger than another conformance constraint ϕ_2 on a subset $H \subseteq \mathbf{Dom}^m$ if $\forall t \in H$, $[\![\phi_1]\!](t) \geq [\![\phi_2]\!](t)$.*

Given a dataset $D \subseteq \mathbf{Dom}^m$ and a projection F, for any tuple t, let $\Delta F(t) = F(t) - \mu(F(D))$. For projections F_1 and F_2, the correlation coefficient ρ_{F_1,F_2} (over D) is defined as $\frac{\frac{1}{|D|}\sum_{t\in D} \Delta F_1(t)\Delta F_2(t)}{\sigma(F_1(D))\sigma(F_2(D))}$.

Definition 9 (Incongruous tuple). *A tuple t is* incongruous *w.r.t. a projection pair $\langle F_1, F_2\rangle$ on D if: $\Delta F_1(t) \cdot \Delta F_2(t) \cdot \rho_{F_1,F_2} < 0$.*

Informally, an incongruous tuple for a pair of projections does not follow the general trend of correlation between the projection pair. For example, if F_1 and F_2 are positively correlated ($\rho_{F_1,F_2} > 0$), an incongruous tuple t deviates in opposite ways from the mean of each projection ($\Delta F_1(t) \cdot \Delta F_2(t) < 0$). Our goal is to find projections that yield a conformance zone with very few incongruous tuples.

Example 10. *In Example 6, X and Y are positively correlated with $\rho_{X,Y} \approx 1$. The tuple $t = (0, 4)$ is incongruous w.r.t. $\langle X, Y\rangle$, because $X(t) = 0 < \mu(X(D)) = 2$, whereas $Y(t) = 4 > \mu(Y(D)) = 2$. Intuitively, the incongruous tuples do not behave like the tuples in D when viewed through the projections X and Y. Note that the narrow conformance zone of Fig. 3(b) no longer contains the incongruous tuple $(0, 4)$. In fact, the conformance zone defined by the conformance constraints derived from projections $X - Y$ and $X + Y$ are free from a vast majority of the incongruous tuples.*

We proceed to state Lemma 11, which informally says that: any two highly-correlated projections can be linearly combined to construct a new projection to obtain a stronger constraint. We write ϕ_F to denote the conformance constraint $\mathtt{lb} \leq F(\vec{A}) \leq \mathtt{ub}$, synthesized from F. (All proofs are in the Appendix.)

Lemma 11. *Let D be a dataset and F_1, F_2 be two projections on D s.t. $|\rho_{F_1,F_2}| \geq \frac{1}{2}$. Then, $\exists\beta_1, \beta_2 \in \mathbb{R}$ s.t. $\beta_1^2 + \beta_2^2 = 1$ and for the new projection $F = \beta_1F_1 + \beta_2F_2$:*
(1) $\sigma(F(D)) < \sigma(F_1(D))$ and $\sigma(F(D)) < \sigma(F_2(D))$, and
(2) ϕ_F is stronger than both ϕ_{F_1} and ϕ_{F_2} on the set of tuples that are incongruous w.r.t. $\langle F_1, F_2\rangle$.

We now extend the result to multiple projections in Theorem 12.

Theorem 12 (Low Standard Deviation Constraints). *Given a dataset D, let $\mathcal{F}=\{F_1, \ldots, F_K\}$ denote a set of projections on D s.t. $\exists F_i, F_j \in \mathcal{F}$ with $|\rho_{F_i,F_j}| \geq \frac{1}{2}$. Then, there exist a nonempty subset $I \subseteq \{1, \ldots, K\}$ and a projection $F = \sum_{k\in I} \beta_k F_k$, where $\beta_k \in \mathbb{R}$ s.t.*

Algorithm 1: Procedure to generate linear projections.

Inputs : A dataset $D \subset \mathbf{Dom}^m$

Output: A set $\{(F_1, \gamma_1), \ldots, (F_K, \gamma_K)\}$ of projections and importance factors

1 $D_N \leftarrow D$ after dropping non-numerical attributes
2 $D'_N \leftarrow [\vec{1}; D_N]$
3 $\{\vec{w}_1, \ldots, \vec{w}_K\} \leftarrow$ eigenvectors of ${D'_N}^T D'_N$
4 **foreach** $1 \leq k \leq K$ **do**
5 $\quad \vec{w}'_k \leftarrow \vec{w}_k$ with first element removed
6 $\quad F_k \leftarrow \lambda \vec{A} : \frac{\vec{A}^T \vec{w}'_k}{||\vec{w}'_k||}$
7 $\quad \gamma_k \leftarrow \frac{1}{\log(2+\sigma(F_k(D_N)))}$
8 **return** $\{(F_1, \frac{\gamma_1}{Z}), \ldots, (F_K, \frac{\gamma_K}{Z})\}$, where $Z = \sum_k \gamma_k$

(1) $\forall k \in I, \sigma(F(D)) < \sigma(F_k(D))$,
(2) $\forall k \in I, \phi_F$ *is stronger than* ϕ_{F_k} *on the subset* H, *where* $H = \{t \mid \forall k \in I(\beta_k \Delta F_k(t) \geq 0) \vee \forall k \in I(\beta_k \Delta F_k(t) \leq 0)\}$, *and*
(3) $\forall k \notin I, |\rho_{F,F_k}| < \frac{1}{2}$.

The theorem establishes that to detect violations for tuples in H: (1) projections with low standard deviations define stronger constraints (and are thus preferable), and (2) a set of constraints with highly-correlated projections is suboptimal (as they can be linearly combined to generate stronger constraints). Note that H is a conservative estimate for the set of tuples where ϕ_F is stronger than each ϕ_{F_k}; there exist tuples outside H for which ϕ_F is stronger.

Bounded projections vs. convex polytope. Bounded projections (Example 7) relate to the computation of convex polytopes [83]. A convex polytope is the smallest possible convex set of tuples that includes all the training tuples; then any tuple falling outside the polytope would be considered non-conforming. The problem with using convex polytopes is that they overfit to the training tuples and is extremely sensitive to outliers. For example, consider a training dataset over attributes X and Y: $\{(1, 10), (2, 20), (3, 30)\}$. A convex polytope in this case would be a line segment—starting at $(1, 10)$ and ending at $(3, 30)$—and the tuple $(5, 50)$ will fall outside it.

Unlike convex polytope—whose goal is to find the smallest possible "inclusion zone" that includes all training tuples—our goal is to find a "conformance zone" that reflects the *trend* of the training tuples. This is inspired from the fact that ML models aim to generalize to tuples outside training set; thus, conformance constraints also need to capture trends and avoid overfitting. Our definition of good conformance constraints (low variance and low mutual correlation) balances overfitting and overgeneralization. Therefore, beyond the minimal bounding hyper-box (i.e., a convex polytope) over the training tuples, we take into consideration the distribution (variance and concentration) of the *interaction* among attributes (trends). For the above example, conformance constraints will model the interaction trend: $Y = 10X$ allowing the tuple $(5, 50)$, which follows the same trend as the training tuples.

4.1.3 PCA-inspired Projection Derivation. Theorem 12 sets the requirements for good projections (see also [51, 56, 84] that make similar observations in different ways). It indicates that we can start with any arbitrary projections and then iteratively improve them. However, we can get the desired set of best projections in one shot using an algorithm inspired by principal component analysis (PCA). PCA relies on computing eigenvectors. There exist different algorithms for computing eigenvectors (from the infinite space of possible vectors). The general mechanism involves applying numerical approaches to iteratively converge to the eigenvectors (up to a desired precision) as no analytical solution exists in general. Our algorithm returns projections that correspond to the principal components of a slightly modified version of the given dataset. Algorithm 1 details our approach for discovering projections for constructing conformance constraints:

Line 1 Drop all non-numerical attributes from D to get the numeric dataset D_N. This is necessary because PCA only applies to numerical values. Instead of dropping, one can also consider embedding techniques to convert non-numerical attributes to numerical ones.

Line 2 Add a new column to D_N that consists of the constant 1, to obtain the modified dataset $D'_N := [\vec{1}; D_N]$, where $\vec{1}$ denotes the column vector with 1 everywhere. We do this transformation to capture the additive constant within principal components, which ensures that the approach works even for unnormalized data.

Line 3 Compute K eigenvectors of the square matrix ${D'_N}^T D'_N$, where K denotes the number of columns in D'_N. These eigenvectors provide coefficients to construct projections.

Lines 5–6 Remove the first element (coefficient for the newly added constant column) of all eigenvectors and normalize them to generate projections. Note that we no longer need the constant element of the eigenvectors since we can appropriately adjust the bounds, lb and ub, for each projection by evaluating it on D_N.

Line 7 Compute importance factor for each projection. Since projections with smaller standard deviations are more discerning (stronger), as discussed in Section 3.2, we assign each projection an importance factor (γ) that is inversely proportional to its standard deviation over D_N.

Line 8 Return the linear projections with corresponding normalized importance factors.

We now claim that the projections returned by Algorithm 1 include the projection with minimum standard deviation and the correlation between any two projections is 0. This indicates that we cannot further improve the projections, and thus they are optimal.

Theorem 13 (Correctness of Algorithm 1). *Given a numerical dataset* D *over the schema* $\mathcal{R}$, *let* $\mathcal{F} = \{F_1, F_2, \ldots, F_K\}$ *be the set of linear projections returned by Algorithm 1. Let* $\sigma^* = \min_k^K \sigma(F_k(D))$. *If* $\mu(A_k(D)) = 0$ *for all attribute* A_k *in* $\mathcal{R}$, *then,*[2]
(1) $\sigma^* \leq \sigma(F(D))\ \forall F = \vec{A}^T \vec{w}$ *where* $||\vec{w}|| \geq 1$, *and*
(2) $\forall F_j, F_k \in \mathcal{F}$ *s.t.* $F_j \neq F_k$, $\rho_{F_j,F_k} = 0$.

Using projections $F_1, \ldots, F_K$, and importance factors $\gamma_1, \ldots, \gamma_K$, returned by Algorithm 1, we generate the simple (conjunctive) constraint with K conjuncts: $\bigwedge_k \mathtt{lb}_k \leq F_k(\vec{A}) \leq \mathtt{ub}_k$. We compute the bounds $\mathtt{lb}_k$ and $\mathtt{ub}_k$ following Section 4.1.1 and use the importance factor γ_k for the k^{th} conjunct in the quantitative semantics.

Example 14. *Algorithm 1 finds the projection of the conformance constraint of Example 1, but in a different form. The actual airlines dataset has an attribute* distance (*DIS*) *that represents miles travelled by a flight. In our experiments, we found the following conformance*

[2]When the condition $\forall A_k\ \mu(A_k(D)) = 0$ does not hold, slightly modified variants of the claim hold. However, by normalizing D (i.e., by subtracting attribute mean $\mu(A_k(D))$ from each $A_k(D)$), it is always possible to satisfy the condition.

constraint[3] *over the dataset of daytime flights:*

$$0.7 \times AT - 0.7 \times DT - 0.14 \times DUR - 0.07 \times DIS \approx 0 \quad (1)$$

This constraint is not quite interpretable by itself, but it is in fact a linear combination of two expected and interpretable constraints: [4]

$$AT - DT - DUR \approx 0 \quad (2)$$

$$DUR - 0.12 \times DIS \approx 0 \quad (3)$$

Here, (2) is the one mentioned in Example 1 and (3) follows from the fact that average aircraft speed is about 500 *mph implying that it requires* 0.12 *minutes per mile.* $0.7 \times$ *(2) +* $0.56 \times$ *(3) yields:*

$$0.7 \times (AT - DT - DUR) + 0.56 \times DUR - 0.56 \times 0.12 \times DIS \approx 0$$
$$\implies 0.7 \times AT - 0.7 \times DT - 0.14 \times DUR - 0.07 \times DIS \approx 0$$

Which is exactly the conformance constraint (1). Algorithm 1 found the optimal projection of (1), which is a linear combination of the projections of (2) and (3). The reason is: there is a correlation between the projections of (2) and (3) over the dataset (Theorem 12). One possible explanation of this correlation is: whenever there is an error in the reported duration of a tuple, it violates both (2) and (3). Due to this natural correlation, Algorithm 1 returned the optimal projection of (1), that "covers" both projections of (2) or (3).

4.2 Compound Conformance Constraints

The quality of our PCA-based simple linear constraints relies on how many low variance linear projections we are able to find on the given dataset. For many datasets, it is possible we find very few, or even none, such linear projections. In these cases, it is fruitful to search for compound constraints; we first focus on *disjunctive constraints* (defined by ψ_A in our language grammar).

PCA-based approach fails in cases where there exist different piecewise linear trends within the data, as it will result into low-quality constraints, with very high variances. In such cases, partitioning the dataset and then learning constraints separately on each partition will result in significant improvement of the learned constraints. A disjunctive constraint is a compound constraint of the form $\bigvee_k((A = c_k) \rhd \phi_k)$, where each ϕ_k is a constraint for a specific partition of D. Finding disjunctive constraints involves horizontally partitioning the dataset D into smaller disjoint datasets $D_1, D_2, \ldots, D_L$. Our strategy for partitioning D is to use categorical attributes with a small domain in D; in our implementation, we use those attributes A_j for which $|\{t.A_j | t \in D\}| \leq 50$. If A_j is such an attribute with values $v_1, v_2, \ldots, v_L$, we partition D into L disjoint datasets $D_1, D_2, \ldots, D_L$, where $D_l = \{t \in D | t.A_j = v_l\}$. Let $\phi_1, \phi_2, \ldots, \phi_L$ be the L simple conformance constraints we learn for $D_1, D_2, \ldots, D_L$ using Algorithm 1, respectively. We compute the following disjunctive conformance constraint for D:

$$((A_j = v_1) \rhd \phi_1) \vee ((A_j = v_2) \rhd \phi_2) \vee \cdots \vee ((A_j = v_L) \rhd \phi_L)$$

We repeat this process and partition D across multiple attributes and generate a compound disjunctive constraint for each attribute. Then we generate the final compound conjunctive conformance constraint (Ψ) for D, which is the conjunction of all these disjunctive constraints. Intuitively, this final conformance constraint forms a set of *overlapping* hyper-boxes around the data tuples.

[3]For ease of exposition, we use $F(\vec{A}) \approx 0$ to denote $\epsilon_1 \leq F(\vec{A}) \leq \epsilon_2$, where $\epsilon_i \approx 0$.
[4]Interpretability is not our explicit goal, but we developed a tool [24] to explain causes of non-conformance. More discussion and case studies are in the Appendix.

4.3 Theoretical Analysis

4.3.1 Runtime Complexity. Computing simple constraints involves two computational steps: (1) computing $X^T X$, where X is an $n \times m$ matrix with n tuples (rows) and m attributes (columns), which takes $O(nm^2)$ time, and (2) computing the eigenvalues and eigenvectors of an $m \times m$ positive definite matrix, which has complexity $O(m^3)$ [58]. Once we obtain the linear projections using the above two steps, we need to compute the mean and variance of these projections on the original dataset, which takes $O(nm^2)$ time. In summary, the overall procedure is cubic in the number of attributes and linear in the number of tuples. For computing disjunctive constraints, we greedily pick attributes that take at most L (typically small) distinct values, and then run the above procedure for simple constraints at most L times. This adds just a constant factor overhead per attribute.

4.3.2 Memory Complexity. The procedure can be implemented in $O(m^2)$ space. The key observation is that $X^T X$ can be computed as $\sum_{i=1}^{n} t_i t_i^T$, where t_i is the i^{th} tuple in the dataset. Thus, $X^T X$ can be computed incrementally by loading only one tuple at a time into memory, computing $t_i t_i^T$, and then adding that to a running sum, which can be stored in $O(m^2)$ space. Note that instead of such an incremental computation, this can also be done in an embarrassingly parallel way where we horizontally partition the data (row-wise) and each partition is computed in parallel.

4.3.3 Implication, Redundancy, and Minimality. Definition 8 gives us the notion of *implication* on conformance constraints: for a dataset D, satisfying ϕ_1 that is stronger than ϕ_2 implies that D would satisfy ϕ_2 as well. Lemma 11 and Theorem 12 associate *redundancy* with correlation: correlated projections can be combined to construct a new projection that makes the correlated projections redundant. Theorem 13 shows that our PCA-based procedure finds a non-redundant (orthogonal and uncorrelated) set of projections. For disjunctive constraints, it is possible to observe redundancy across partitions. However, our quantitative semantics ensures that redundancy does not affect the violation score. Another notion relevant to data profiles (e.g., FDs) is *minimality*. In this work, we do not focus on finding the minimal set of conformance constraints. Towards achieving minimality for conformance constraints, a future direction is to explore techniques for optimal data partitioning. However, our approach computes only m conformance constraints for each partition. Further, for a single tuple, only $m_N \cdot m_C$ conformance constraints are applicable, where m_N and m_C are the number of numerical and categorical attributes in D (i.e., $m = m_N + m_C$). The quantity $m_N \cdot m_C$ is upper-bounded by $\frac{m^2}{4}$.

5 TRUSTED MACHINE LEARNING

In this section, we provide a theoretical justification of why conformance constraints are effective in identifying tuples for which learned models are likely to make incorrect predictions. To that end, we define *unsafe* tuples and show that an "ideal" conformance constraint provides a sound and complete mechanism to detect unsafe tuples. In Section 4, we showed that low-variance projections construct strong conformance constraints, which yield a small conformance zone. We now make a similar argument, but in a slightly different way: we show that projections with zero variance give us

equality constraints that are useful for trusted machine learning. We start with an example to provide the intuition.

Example 15. *Consider the airlines dataset D and assume that all tuples in D satisfy the equality constraint $\phi := AT - DT - DUR = 0$ (i.e.,* $\mathtt{lb} = \mathtt{ub} = 0$*). Note that for equality constraint, the corresponding projection has* zero *variance—the lowest possible variance. Now, suppose that the task is to learn some function $f(AT, DT, DUR)$. If the above constraint holds for D, then the ML model can instead learn the function $g(AT, DT, DUR) = f(DT + DUR, DT, DUR)$. g will perform just as well as f on D: in fact, it will produce the same output as f on D. If a new serving tuple t satisfies ϕ, then $g(t) = f(t)$, and the prediction will be correct. However, if t does not satisfy ϕ, then $g(t)$ will likely be significantly different from $f(t)$. Hence, violation of the conformance constraint is a strong indicator of performance degradation of the learned prediction model. Note that f need not be a linear function: as long as g is also in the class of models that the learning procedure is searching over, the above argument holds.*

Based on the intuition of Example 15, we proceed to formally define unsafe tuples. We use $[D; Y]$ to denote the *annotated dataset* obtained by appending the target attribute Y to a dataset D, and $\mathtt{coDom}$ to denote Y's domain.

Definition 16 (Unsafe tuple). *Given a class $\mathcal{C}$ of functions with signature $\mathbf{Dom}^m \mapsto \mathtt{coDom}$, and an annotated dataset $[D; Y] \subset (\mathbf{Dom}^m \times \mathtt{coDom})$, a tuple $t \in \mathbf{Dom}^m$ is unsafe w.r.t. $\mathcal{C}$ and $[D; Y]$, if $\exists f, g \in \mathcal{C}$ s.t. $f(D) = g(D) = Y$ but $f(t) \neq g(t)$.*

Intuitively, t is unsafe if there exist two different predictor functions f and g that agree on all tuples in D, but disagree on t. Since, we can never be sure whether the model learned f or g, we should be cautious about the prediction on t. Example 15 suggests that t can be unsafe when all tuples in D satisfy the equality conformance constraint $f(\vec{A}) - g(\vec{A}) = 0$ but t does not. Hence, we can use the following approach for trusted machine learning:

(1) Learn conformance constraints Φ for the dataset D.
(2) Declare t as unsafe if t does not satisfy Φ.

The above approach is sound and complete for characterizing unsafe tuples, thanks to the following proposition.

Proposition 17. *There exists a conformance constraint Φ for D s.t. the following statement is true: "$\neg\Phi(t)$ iff t is unsafe w.r.t. $\mathcal{C}$ and $[D; Y]$ for all $t \in \mathbf{Dom}^m$".*

The required conformance constraint Φ is: $\forall f, g \in \mathcal{C} : f(D) = g(D) = Y \implies f(\vec{A}) - g(\vec{A}) = 0$. Intuitively, when all possible pairs of functions that agree on D also agree on t, only then the prediction on t can be trusted. (More discussion is in the Appendix.)

5.1 Applicability

Generalization to noisy setting. While our analysis and formalization for using conformance constraints for TML focused on the noise-free setting, the intuition generalizes to noisy data. Specifically, suppose that f and g are two possible functions a model may learn over D; then, we expect that the difference $f - g$ will have small variance over D, and thus would be a good conformance constraint. In turn, the violation of this constraint would mean that f and g diverge on a tuple t (making t unsafe); since we are oblivious of the function the model learned, prediction on t is untrustworthy.

False positives. Conformance constraints are designed to work in a model-agnostic setting. Although this setting is of great practical importance, designing a perfect mechanism for quantifying trust in ML model predictions, while remaining completely model-agnostic, is challenging. It raises the concern of *false positives*: conformance constraints may incorrectly flag tuples for which the model's prediction is in fact correct. This may happen when the model ignores the trend that conformance constraints learn. Since we are oblivious of the prediction task and the model, it is preferable that conformance constraints behave rather *conservatively* so that the users can be cautious about potentially unsafe tuples. Moreover, if a model ignores some attributes (or their interactions) during training, it is still necessary to learn conformance constraints over them. Particularly, in case of concept drift [81], the ground truth may start depending on those attributes, and by learning conformance constraints over all attributes, we can better detect potential model failures.

False negatives. Another concern involving conformance constraints is of *false negatives*: linear conformance constraints may miss nonlinear constraints, and thus fail to identify some unsafe tuples. However, the linear dependencies modeled in conformance constraints persist even after sophisticated (nonlinear) attribute transformations. Therefore, violation of conformance constraints is a strong indicator of potential failure of a possibly nonlinear model.

Modeling nonlinear constraints. While linear conformance constraints are the most common ones, we note that our framework can be easily extended to support nonlinear conformance constraints using *kernel functions* [69]—which offer an efficient, scalable, and powerful mechanism to learn nonlinear decision boundaries for support vector machines (also known as "kernel trick"). Briefly, instead of explicitly augmenting the dataset with transformed nonlinear attributes—which grows exponentially with the desired degree of polynomials—kernel functions enable *implicit* search for nonlinear models. The same idea also applies for PCA called kernel-PCA [9, 41]. While we limit our evaluation to only linear kernel, polynomial kernels—e.g., radial basis function (RBF) [45]—can be plugged into our framework to model nonlinear conformance constraints.

In general, our conformance language is not guaranteed to model all possible functions that an ML model can potentially learn, and thus is not guaranteed to find the best conformance constraint. However, our empirical evaluation on real-world datasets shows that our language models conformance constraints effectively.

6 EXPERIMENTAL EVALUATION

We now present experimental evaluation to demonstrate the effectiveness of conformance constraints over our two case-study applications (Section 2): trusted machine learning and data drift. Our experiments target the following research questions:

- How effective are conformance constraints for trusted machine learning? Is there a relationship between constraint violation score and the ML model's prediction accuracy? (Section 6.1)
- Can conformance constraints be used to quantify data drift? How do they compare to other state-of-the-art drift-detection techniques? (Section 6.2)

Efficiency. In all our experiments, our algorithms for deriving conformance constraints were extremely fast, and took only a few seconds even for datasets with 6 million rows. The number of

attributes were reasonably small (~40), which is true for most practical applications. As our theoretical analysis showed (Section 4.3), our approach is linear in the number of data rows and cubic in the number of attributes. Since the runtime performance of our techniques is straightforward, we opted to not include further discussion of efficiency here and instead focus this empirical analysis on the techniques' effectiveness.

Implementation: CCSynth. We created an open-source implementation of conformance constraints and our method for synthesizing them, CCSynth, in Python 3. Experiments were run on a Windows 10 machine (3.60 GHz processor and 16GB RAM).

Datasets

Airlines [8] contains data about flights and has 14 attributes —year, month, day, day of week, departure time, arrival time, carrier, flight number, elapsed time, origin, destination, distance, diverted, and arrival delay. We used a subset of the data containing all flight information for year 2008. In this dataset, most of the attributes follow uniform distribution (e.g., month, day, arrival and departure time, etc.); elapsed time and distance follow skewed distribution with higher concentration towards small values (implying that shorter flights are more common); arrival delay follows a slightly skewed gaussian distribution implying most flights are on-time, few arrive late and even fewer arrive early. The training and serving datasets contain 5.4M and 0.4M rows, respectively.

Human Activity Recognition (HAR) [78] is a real-world dataset about activities for 15 individuals, 8 males and 7 females, with varying fitness levels and BMIs. We use data from two sensors—accelerometer and gyroscope—attached to 6 body locations—head, shin, thigh, upper arm, waist, and chest. We consider 5 activities—lying, running, sitting, standing, and walking. The dataset contains 36 numerical attributes (2 sensors × 6 body-locations × 3 coordinates) and 2 categorical attributes—activity-type and person-ID. We pre-processed the dataset to aggregate the measurements over a small time window, resulting in 10,000 tuples per person and activity, for a total of 750,000 tuples.

Extreme Verification Latency (EVL) [74] is a widely used benchmark to evaluate drift-detection algorithms in non-stationary environments under extreme verification latency. It contains 16 synthetic datasets with incremental and gradual concept drifts over time. The number of attributes of these datasets vary from 2 to 6 and each of them has one categorical attribute.

6.1 Trusted Machine Learning

We now demonstrate the applicability of conformance constraints in the TML problem. We show that, serving tuples that violate the training data's conformance constraints are unsafe, and therefore, an ML model is more likely to perform poorly on those tuples.

Airlines. We design a regression task of predicting the arrival delay and train a linear regression model for the task. Our goal is to observe whether the mean absolute error of the predictions (positively) correlates to the constraint violation for the serving tuples. In a process analogous to the one described in Example 1, our training dataset (`Train`) comprises of a subset of daytime flights—flights that have arrival time later than the departure time (in 24 hour format). We design three serving sets: (1) `Daytime`: similar to `Train`, but another subset, (2) `Overnight`: flights that have arrival time earlier than the departure time (the dataset does not explicitly report the date of arrival), and (3) `Mixed`: a mixture of `Daytime` and `Overnight`. A few sample tuples of this dataset are in Fig. 1.

	Train	Serving		
		Daytime	Overnight	Mixed
Average violation	0.02%	0.02%	27.68%	8.87%
MAE	18.95	18.89	80.54	38.60

Figure 4: Average constraint violation (in percentage) and MAE (for linear regression) of four data splits on the airlines dataset. The constraints were learned on `Train`, excluding the target attribute, `delay`.

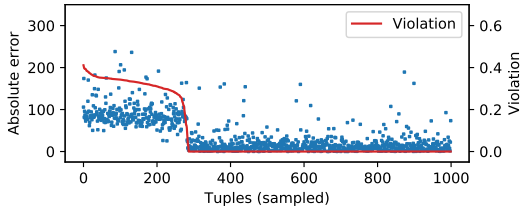


Figure 5: Constraint violation strongly correlates with the absolute error of delay prediction of a linear regression model.

Our experiment involves the following steps: (1) CCSynth computes conformance constraints Φ over `Train`, while *ignoring* the target attribute `delay`. (2) We compute average constraint violation for all four datasets—`Train`, `Daytime`, `Overnight`, and `Mixed`—against Φ (first row of Fig. 4). (3) We train a linear regression model over `Train`—including `delay`—that learns to predict arrival delay. (4) We compute mean absolute error (MAE) of the prediction accuracy of the regressor over the four datasets (second row of Fig. 4). We find that constraint violation is a very good proxy for prediction error, as they vary in a similar manner across the four datasets. The reason is that the model implicitly assumes that the constraints (e.g., $AT - DT - DUR \approx 0$) derived by CCSynth will always hold, and, thus, deteriorates when the assumption no longer holds.

To observe the rate of false positives and false negatives, we investigate the relationship between constraint violation and prediction error at tuple-level granularity. We sample 1000 tuples from `Mixed` and organize them by decreasing order of violations (Fig. 5). For all the tuples (on the left) that incur high constraint violations, the regression model incurs high error for them as well. This implies that CCSynth reports no false positives. There are some false negatives (right part of the graph), where violation is low, but the prediction error is high. Nevertheless, such false negatives are very few.

HAR. On the HAR dataset, we design a supervised classification task to identify persons from their activity data that contains 36 numerical attributes. We construct `train_x` with data for sedentary activities (lying, standing, and sitting), and `train_y` with the corresponding person-IDs. We learn conformance constraints on `train_x`, and train a Logistic Regression (LR) classifier using the annotated dataset [`train_x`; `train_y`]. During serving, we mix mobile activities (walking and running) with held-out data for sedentary activities and observe how the classification's mean accuracy-drop (i.e., how much the mean prediction accuracy decreases compared to the mean prediction accuracy over the training data) relates to average constraint violation. To avoid any artifact due to sampling bias, we repeat this experiment 10 times for different subsets of the data

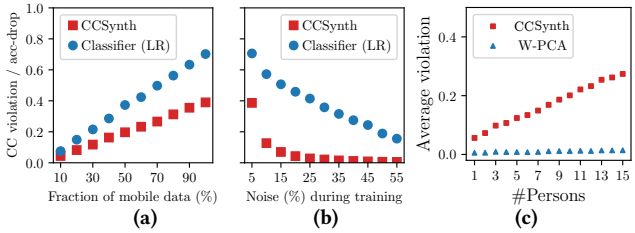


Figure 6: (a) As a higher fraction of mobile activity data is mixed with sedentary activity data, conformance constraints are violated more, and the classifier's mean accuracy-drop increases. (b) As more noise is added during training, conformance constraints get weaker, leading to less violation and decreased accuracy-drop. (c) CCSynth detects the gradual local drift on the HAR dataset as more people start changing their activities. In contrast, weighted-PCA (W-PCA) fails to detect drift in absence of a strong global drift.

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	Fitness	BMI	Gender
p1	0	0.3	0.4	0.3	0.3	0.3	0.3	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.3	Moderate	Underweight	Female
p2	0.4	0	0.4	0.3	0.3	0.3	0.3	0.5	0.3	0.3	0.4	0.3	0.4	0.4	0.3	Moderate	Normal	Male
p3	0.5	0.5	0	0.4	0.5	0.5	0.5	0.6	0.5	0.5	0.5	0.5	0.5	0.5	0.5	Moderate	Overweight	Male
p4	0.2	0.2	0.3	0	0.3	0.2	0.3	0.5	0.2	0.3	0.2	0.3	0.3	0.3	0.3	Moderate	Normal	Male
p5	0.2	0.3	0.4	0.2	0	0.2	0.2	0.4	0.2	0.3	0.3	0.3	0.3	0.3	0.3	Moderate	Normal	Male
p6	0.3	0.3	0.4	0.3	0.3	0	0.3	0.4	0.2	0.2	0.2	0.2	0.3	0.2	0.4	High	Normal	Female
p7	0.3	0.3	0.4	0.3	0.3	0.3	0	0.4	0.3	0.2	0.3	0.3	0.3	0.3	0.3	Moderate	Overweight	Male
p8	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.4	0.5	0.4	0.5	0.5	0.5	0.5	Low	Obese	Female
p9	0.4	0.4	0.5	0.4	0.3	0.3	0.5	0.5	0	0.3	0.4	0.4	0.4	0.4	0.5	High	Overweight	Male
p10	0.4	0.3	0.4	0.3	0.3	0.3	0.4	0.5	0.3	0	0.3	0.3	0.3	0.4	0.3	Moderate	Obese	Male
p11	0.4	0.4	0.5	0.3	0.4	0.3	0.4	0.5	0.3	0.3	0	0.3	0.3	0.3	0.4	Moderate	Normal	Female
p12	0.3	0.3	0.4	0.3	0.3	0.3	0.3	0.5	0.3	0.3	0.3	0	0.3	0.4	0.4	Moderate	Normal	Female
p13	0.3	0.3	0.4	0.3	0.3	0.2	0.3	0.4	0.2	0.3	0.2	0.3	0	0.3	0.4	Moderate	Normal	Female
p14	0.3	0.3	0.4	0.3	0.3	0.2	0.3	0.4	0.2	0.3	0.3	0.3	0.3	0	0.3	High	Normal	Male
p15	0.4	0.4	0.5	0.4	0.4	0.5	0.4	0.5	0.5	0.4	0.4	0.4	0.5	0.5	0	Low	Normal	Female

Figure 7: Inter-person constraint violation heat map. Each person has a very low self-violation.

by randomly sampling 5000 data points for each of training and serving. Fig. 6(a) depicts our findings: classification degradation has a clear positive correlation with violation (pcc = 0.99 with p-value = 0).

Noise sensitivity. Intuitively, noise weakens conformance constraints by increasing variance in the training data, which results in reduced violations of the serving data. However, this is desirable: as more noise makes machine-learned models less likely to overfit, and, thus, more robust. In our experiment for observing noise sensitivity of conformance constraints, we use only mobile activity data as the serving set and start with sedentary data as the training set. Then we gradually introduce noise in the training set by mixing mobile activity data. As Fig. 6(b) shows, when more noise is added to the training data, conformance constraints start getting weaker; this leads to reduction in violations. However, the classifier also becomes robust with more noise, which is evident from gradual decrease in accuracy-drop (i.e., increase in accuracy). Therefore, even under the presence of noise, the positive correlation between classification degradation and violation persists (pcc = 0.82 with p-value = 0.002).

Key takeaway: CCSynth derives conformance constraints whose violation is a strong proxy of model prediction accuracy. Their correlation persists even in the presence of noise.

6.2 Data Drift

We now present results of using conformance constraints for drift-detection; specifically, for *quantifying* drift in data. Given a baseline dataset D, and a new dataset D', the drift is measured as average violation of tuples in D' on conformance constraints learned for D.

HAR. We perform two drift-quantification experiments on HAR:

Gradual drift. For observing how CCSynth detects gradual drift, we introduce drift in an organic way. The initial training dataset contains data of exactly one activity for each person. This is a realistic scenario as one can think of it as taking a snapshot of what a group of people are doing during a reasonably small time window. We introduce gradual drift to the initial dataset by altering the activity of one person at a time. To control the amount of drift, we use a parameter K. When $K = 1$, the first person switches their activity, i.e., we replace the tuples corresponding to the first person performing activity A with new tuples that correspond to the same person performing another activity B. When $K = 2$, the second person switches their activity in a similar fashion, and so on. As we increase K from 1 to 15, we expect a gradual increase in the drift magnitude compared to the initial training data. When $K = 15$, all persons have switched their activities from the initial setting, and we expect to observe maximum drift. We repeat this experiment 10 times, and display the average constraint violation in Fig. 6(c): the drift magnitude (violation) indeed increases as more people alter their activities.

The baseline weighted-PCA approach (W-PCA) fails to model local constraints (who is doing what), and learns some weaker global constraints indicating that "a group of people are performing some activities". Thus, it fails to detect the gradual local drift. CCSynth can detect drift when individuals switch activities, as it learns *disjunctive* constraints that encode who is doing what.

Inter-person drift. The goal of this experiment is to observe how effectively conformance constraints can model the representation of an entity and whether such learned representations can be used to accurately quantify drift between two entities. We use half of each person's data to learn the constraints, and compute violation on the held-out data. CCSynth learns disjunctive constraints for each person over all activities, and then we use the violation w.r.t. the learned constraints to measure how much the other persons drift. While computing drift between two persons, we compute activity-wise constraint violation scores and then average them out. In Fig. 7, the violation score at row p1 and column p2 denotes how much p2 drifts from p1. As one would expect, we observe a very low self-drift across the diagonal. Interestingly, our result also shows that some people are more different from others, which appears to have some correlation with (the hidden ground truth) fitness and BMI values. This asserts that the constraints we learn for each person are an accurate abstraction of that person's activities, as people do not deviate too much from their usual activity patterns.

EVL. We now compare CCSynth against other state-of-the-art drift detection approaches on the EVL benchmark.

Baselines. We use two drift-detection baselines as described below:

(1) PCA-SPLL [51][5], similar to us, also argues that principal components with lower variance are more sensitive to a general drift, and uses those for dimensionality reduction. It then models

[5] SPLL source code: github.com/LucyKuncheva/Change-detection

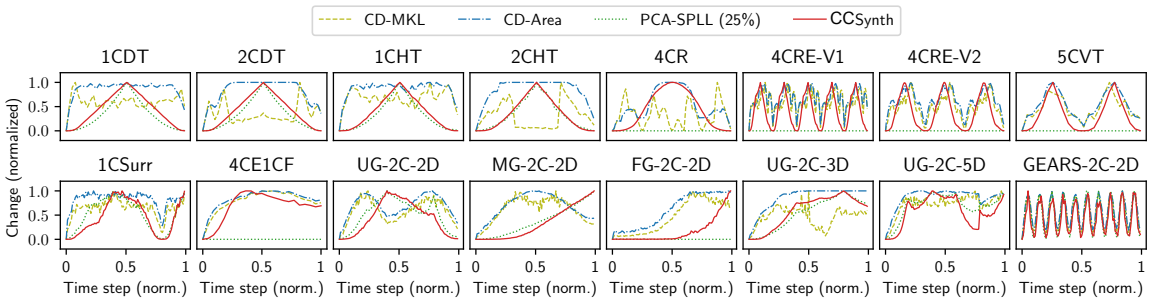


Figure 8: In the EVL benchmark, CCSynth quantifies drift correctly for all cases, outperforming other approaches. PCA-SPLL fails to detect drift in a few cases by discarding all principal components; CD-MKL and CD-Area are too sensitive to small drift and detect spurious drifts.

multivariate distribution over the reduced dimensions and applies semi-parametric log-likelihood (SPLL) to detect drift between two multivariate distributions. However, PCA-SPLL discards all high-variance principal components and does not model disjunctive constraints.

(2) CD (Change Detection) [63][6] is another PCA-based approach for drift detection in data streams. But unlike PCA-SPLL, it ignores low-variance principal components. CD projects the data onto top k high-variance principal components, which results into multiple univariate distributions. We compare against two variants of CD: CD-Area, which uses the intersection area under the curves of two density functions as a divergence metric, and CD-MKL, which uses Maximum KL-divergence as a symmetric divergence metric, to compute divergence between the univariate distributions.

Fig. 8 depicts how CCSynth compares against CD-MKL, CD-Area, and PCA-SPLL, on 16 datasets in the EVL benchmark. For PCA-SPLL, we retain principal components that contribute to a cumulative explained variance below 25%. Beyond drift detection, which just detects if drift is above some threshold, we focus on drift quantification. A tuple (x, y) in the plots denotes that drift magnitude for dataset at x^{th} time window, w.r.t. the dataset at the first time window, is y. Since different approaches report drift magnitudes in different scales, we normalize the drift values within [0, 1]. Additionally, since different datasets have different number of time windows, for the ease of exposition, we normalize the time window indices. Below we state our key findings from this experiment:

CCSynth's drift quantification matches the ground truth. In all of the datasets in the EVL benchmark, CCSynth is able to correctly quantify the drift, which matches the ground truth [2] exceptionally well. In contrast, as CD focuses on detecting the drift point, it is ill-equipped to precisely quantify the drift, which is demonstrated in several cases (e.g., 2CHT), where CD fails to distinguish the deviation in drift magnitudes. In contrast, both PCA-SPLL and CCSynth correctly quantify the drift. Since CD only retains high-variance principal components, it is more susceptible to noise and considers noise in the dataset as significant drift, which leads to incorrect drift quantification. In contrast, PCA-SPLL and CCSynth ignore the noise and only capture the general notion of drift. In all of the EVL datasets, we found CD-Area to work better than CD-MKL, which also agrees with the authors' experiments.

[6]CD source code: mine.kaust.edu.sa/Pages/Software.aspx

CCSynth models local drift. When the dataset contains instances from multiple classes, the drift may be just local, and not global (e.g., 4CR dataset as shown in the Appendix). In such cases, PCA-SPLL fails to detect drift (4CR, 4CRE-V2, and FG-2C-2D). In contrast, CCSynth learns disjunctive constraints and quantifies local drifts accurately.

Key takeaways: CCSynth can effectively detect data drift, both global and local, is robust across drift patterns, and significantly outperforms the state-of-the-art methods.

7 RELATED WORK

There is extensive literature on data-profiling [5] primitives that model relationships among data attributes, such as functional dependencies (FD) [59, 93] and their variants (metric [48], conditional [23], soft [38], approximate [36, 50], relaxed [16], pattern [62], etc.), differential dependencies [72], denial constraints [13, 17, 53, 61], statistical constraints [91], etc. However, none of them focus on learning approximate arithmetic relationships that involve multiple numerical attributes in a noisy setting, which is the focus of our work. Some variants of FDs [16, 36, 38, 48, 50] consider noisy setting, but they require noise parameters to be explicitly specified by the user. In contrast, we do not require any explicit noise parameter.

The issue of trust, resilience, and interpretability of artificial intelligence (AI) systems has been a theme of increasing interest recently [39, 42, 67, 86], particularly for safety-critical data-driven AI systems [80, 87]. A standard way to decide whether to trust a classifier or not, is to use the classifier-produced confidence score. However, as a prior work [41] argues, this is not always effective since the classifier's confidence scores are not well-calibrated. While some recent techniques [20, 31, 41, 68] aim at validating the inferences made by machine-learned models on unseen tuples, they usually require knowledge of the inference task, access to the model, and/or expected cases of data shift, which we do not. Furthermore, they usually require costly hyper-parameter tuning and do not generate closed-form data profiles like conformance constraints (Fig. 2). Prior work on data drift, change detection, and covariate shift [7, 11, 14, 18, 19, 21, 22, 25, 26, 33, 34, 37, 44, 46, 66, 70, 71, 73, 82, 88] relies on modeling data distribution. However, data distribution does not capture constraints, which is the primary focus of our work.

Few works [20, 31, 54] use autoencoder's [32, 65] input reconstruction error to determine if a new data point is out-of-distribution.

Our approach is similar to outlier-detection [49] and one-class-classification [79]. However, conformance constraints differ from these approaches as they perform under the additional requirement to generalize the data in a way that is exploited by a given class of ML models. In general, there is a clear gap between representation learning (that models data likelihood) [6, 32, 43, 65] and the (constraint-oriented) data-profiling techniques to address the problem of trusted AI and our aim is to bridge this gap.

8 SUMMARY AND FUTURE DIRECTIONS

We introduced conformance constraints, and the notion of unsafe tuples for trusted machine learning. We presented an efficient and highly-scalable approach for synthesizing conformance constraints; and demonstrated their effectiveness to tag unsafe tuples and quantify data drift. The experiments validate our theory and our principle of using low-variance projections to generate effective conformance constraints. We have studied only two use-cases from a large pool of potential applications using linear conformance constraints. In future, we want to explore more powerful nonlinear conformance constraints using autoencoders. Moreover, we plan to explore approaches to learn conformance constraints in a decision-tree-like structure where categorical attributes will guide the splitting conditions and leaves will contain simple conformance constraints. Further, we envision a mechanism—built on top of conformance constraints—to explore differences between datasets.

REFERENCES

[1] Cardiovascular disease: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset, Feb 2020.

[2] Extreme verification latency benchmark video (nonstationary environments - archive): sites.google.com/site/nonstationaryarchive/home, Feb 2020.

[3] House prices: Advanced regression techniques: Advanced regression techniques. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data, Feb 2020.

[4] Mobile prices: https://www.kaggle.com/iabhishekofficial/mobile-price-classification, Feb 2020.

[5] Z. Abedjan, L. Golab, and F. Naumann. Profiling relational data: a survey. *VLDB J.*, 24(4):557–581, 2015.

[6] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

[7] C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *SIGMOD*, pages 575–586, 2003.

[8] Airlines dataset., 2019. http://kt.ijs.si/elena_ikonomovska/data.html.

[9] C. Alzate and J. A. Suykens. Kernel component analysis using an epsilon-insensitive robust loss function. *IEEE Transactions on Neural Networks*, 19(9):1583–1598, 2008.

[10] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer. A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, 127:278–294, 2017.

[11] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *SDM*, pages 443–448, 2007.

[12] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010.

[13] T. Bleifuß, S. Kruse, and F. Naumann. Efficient denial constraint discovery with hydra. *PVLDB*, 11(3):311–323, 2017.

[14] L. Bu, C. Alippi, and D. Zhao. A pdf-free change detection test based on density difference estimation. *IEEE Trans. Neural Netw. Learning Syst.*, 29(2):324–334, 2018.

[15] G. Buehrer, B. W. Weide, and P. A. G. Sivilotti. Using parse tree validation to prevent SQL injection attacks. In *International Workshop on Software Engineering and Middleware, SEM*, pages 106–113, 2005.

[16] L. Caruccio, V. Deufemia, and G. Polese. On the discovery of relaxed functional dependencies. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, pages 53–61, 2016.

[17] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013.

[18] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Symp. on the Interface of Statistics, Computing Science, and Applications*, 2006.

[19] R. F. de Mello, Y. Vaz, C. H. G. Ferreira, and A. Bifet. On learning guarantees to unsupervised concept drift detection on data streams. *Expert Syst. Appl.*, 117:90–102, 2019.

[20] T. Denouden, R. Salay, K. Czarnecki, V. Abdelzad, B. Phan, and S. Vernekar. Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance. *CoRR*, abs/1812.02765, 2018.

[21] D. M. dos Reis, P. A. Flach, S. Matwin, and G. E. A. P. A. Batista. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In *SIGKDD*, pages 1545–1554, 2016.

[22] W. J. Faithfull, J. J. R. Diez, and L. I. Kuncheva. Combining univariate approaches for ensemble change detection in multivariate data. *Information Fusion*, 45:202–214, 2019.

[23] W. Fan, F. Geerts, L. V. S. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In *ICDE*, pages 1231–1234, 2009.

[24] A. Fariha, A. Tiwari, A. Radhakrishna, and S. Gulwani. Extune: Explaining tuple non-conformance. In *SIGMOD*, pages 2741–2744, 2020.

[25] M. M. Gaber and P. S. Yu. Classification of changes in evolving data streams using online clustering result deviation. In *Proc. Of International Workshop on Knowledge Discovery in Data Streams*, 2006.

[26] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence - SBIA*, pages 286–295, 2004.

[27] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[28] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330, 2017.

[29] J. H. Hayes and A. J. Offutt. Increased software reliability through input validation analysis and testing. In *International Symposium on Software Reliability Engineering, ISSRE*, pages 199–209, 1999.

[30] A. Heise, J. Quiané-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann. Scalable discovery of unique column combinations. *PVLDB*, 7(4):301–312, 2013.

[31] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

[32] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[33] S. Ho. A martingale framework for concept change detection in time-varying data streams. In *ICML*, pages 321–327, 2005.

[34] B. Hooi and C. Faloutsos. Branch and border: Partition-based change detection in multivariate time series. In *SDM*, pages 504–512, 2019.

[35] R. A. Horn and C. R. Johnson. *Matrix Analysis*. New York, NY, USA, 2nd edition, 2012.

[36] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999.

[37] D. Ienco, A. Bifet, B. Pfahringer, and P. Poncelet. Change detection in categorical evolving data streams. In *SAC*, pages 792–797, 2014.

[38] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga. CORDS: automatic discovery of correlations and soft functional dependencies. In *SIGMOD*, pages 647–658, 2004.

[39] S. Jha. Trust, resilience and interpretability of AI models. In *Numerical Software Verification - 12th International Workshop, NSV@CAV*, pages 3–25, 2019.

[40] H. Jiang, S. G. Elbaum, and C. Detweiler. Inferring and monitoring invariants in robotic systems. *Auton. Robots*, 41(4):1027–1046, 2017.

[41] H. Jiang, B. Kim, M. Y. Guan, and M. R. Gupta. To trust or not to trust A classifier. In *NeurIPS*, pages 5546–5557, 2018.

[42] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia. Model assertions for monitoring and improving ML models. In *MLSys*, 2020.

[43] T. Karaletsos, S. Belongie, and G. Rätsch. Bayesian representation learning with oracle constraints. *arXiv preprint arXiv:1506.05011*, 2015.

[44] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *SDM*, pages 389–400, 2009.

[45] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[46] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *PVLDB*, pages 180–191, 2004.

[47] R. Koch. Sql database performance tuning for developers. https://moa.cms.waikato.ac.nz/datasets/, Sep 2013.

[48] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. Metric functional dependencies. In *ICDE*, pages 1275–1278, 2009.

[49] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Outlier detection in arbitrarily oriented subspaces. In *2012 IEEE 12th international conference on data mining*, pages 379–388, 2012.

[50] S. Kruse and F. Naumann. Efficient discovery of approximate dependencies. *PVLDB*, 11(7):759–772, 2018.

[51] L. I. Kuncheva and W. J. Faithfull. PCA feature extraction for change detection in multidimensional unlabeled data. *IEEE Trans. Neural Netw. Learning Syst.*, 25(1):69–80, 2014.

[52] P. Langer and F. Naumann. Efficient order dependency detection. *VLDB J.*, 25(2):223–241, 2016.

[53] E. Livshits, A. Heidari, I. F. Ilyas, and B. Kimelfeld. Approximate denial constraints. *CoRR*, abs/2005.08540, 2020.

[54] H. Lu, H. Xu, N. Liu, Y. Zhou, and X. Wang. Data sanity check for deep learning systems via learnt assertions. *CoRR*, abs/1909.03835, 2019.

[55] F. D. Marchi, S. Lopes, and J. Petit. Unary and n-ary inclusion dependency discovery in relational databases. *J. Intell. Inf. Syst.*, 32(1):53–73, 2009.

[56] T. Martin and I. K. Glad. Online detection of sparse changes in high-dimensional data streams using tailored projections. *arXiv preprint arXiv:1908.02029*, 2019.

[57] Z. Ouyang, Y. Gao, Z. Zhao, and T. Wang. Study on the classification of data streams with concept drift. In *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 3, pages 1673–1677, 2011.

[58] V. Y. Pan and Z. Q. Chen. The complexity of the matrix eigenproblem. In *ACM Symposium on Theory of Computing*, pages 507–516, 1999.

[59] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J.-P. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *PVLDB*, 8(10):1082–1093, 2015.

[60] T. Papenbrock, S. Kruse, J.-A. Quiané-Ruiz, and F. Naumann. Divide & conquer-based inclusion dependency discovery. *PVLDB*, 8(7):774–785, 2015.

[61] E. H. Pena, E. C. de Almeida, and F. Naumann. Discovery of approximate (and exact) denial constraints. *PVLDB*, 13(3):266–278, 2019.

[62] A. Qahtan, N. Tang, M. Ouzzani, Y. Cao, and M. Stonebraker. Pattern functional dependencies for data cleaning. *PVLDB*, 13(5):684–697, 2020.

[63] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *SIGKDD*, pages 935–944, 2015.

[64] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *SIGKDD*, pages 1135–1144, 2016.

[65] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[66] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda. A new method for data stream mining based on the misclassification error. *IEEE Trans. Neural Netw. Learning Syst.*, 26(5):1048–1059, 2015.

[67] S. Saria and A. Subbaswamy. Tutorial: Safe and reliable machine learning. *CoRR*, abs/1904.07204, 2019.

[68] S. Schelter, T. Rukat, and F. Bießmann. Learning to validate the predictions of black box classifiers on unseen data. In *SIGMOD*, pages 1289–1299, 2020.

[69] B. Schölkopf, A. J. Smola, F. Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 2002.

[70] T. S. Sethi and M. M. Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Syst. Appl.*, 82:77–99, 2017.

[71] T. S. Sethi, M. M. Kantardzic, and E. Arabmakki. Monitoring classification blindspots to detect drifts from unlabeled data. In *IEEE International Conference on Information Reuse and Integration, IRI*, pages 142–151, 2016.

[72] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *ACM Transactions on Database Systems (TODS)*, 36(3):1–41, 2011.

[73] X. Song, M. Wu, C. M. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *SIGKDD*, pages 667–676, 2007.

[74] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In *SDM*, pages 873–881, 2015.

[75] A. Subbaswamy, P. Schulam, and S. Saria. Preventing failures due to dataset shift: Learning predictive models that transport. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 3118–3127, 2019.

[76] C. A. Sutton, T. Hobson, J. Geddes, and R. Caruana. Data diff: Interpretable, executable summaries of changes in distributions for data wrangling. In *SIGKDD*, pages 2279–2288, 2018.

[77] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. Effective and complete discovery of order dependencies via set-based axiomatization. *PVLDB*, 10(7):721–732, 2017.

[78] T. Sztyler and H. Stuckenschmidt. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *PerCom*, pages 1–9, 2016.

[79] D. M. J. Tax and K. Müller. Feature extraction for one-class classification. In *ICANN/ICONIP*, pages 342–349, 2003.

[80] A. Tiwari, B. Dutertre, D. Jovanovic, T. de Candia, P. Lincoln, J. M. Rushby, D. Sadigh, and S. A. Seshia. Safety envelope for security. In *HiCoNS*, pages 85–94, 2014.

[81] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

[82] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections. In *IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, pages 679–684, 2006.

[83] F. Turchini, L. Seidenari, and A. Del Bimbo. Convex polytope ensembles for spatio-temporal anomaly detection. In *International Conference on Image Analysis and Processing*, pages 174–184, 2017.

[84] M. Tveten. Which principal components are most sensitive to distributional changes? *arXiv preprint arXiv:1905.06318*, 2019.

[85] V. Vapnik, S. E. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation and signal processing. In *NeurIPS*, pages 281–287, 1997.

[86] K. R. Varshney. Trustworthy machine learning and artificial intelligence. *ACM Crossroads*, 25(3):26–29, 2019.

[87] K. R. Varshney and H. Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big Data*, 5(3):246–255, 2017.

[88] H. Wang and Z. Abraham. Concept drift detection for imbalanced stream data. *CoRR*, abs/1504.01044, 2015.

[89] G. Wassermann and Z. Su. Sound and precise analysis of web applications for injection vulnerabilities. In *PLDI*, pages 32–41, 2007.

[90] L. White and E. I. Cohen. A domain strategy for computer program testing. *IEEE Trans. Software Engineering*, SE-6(3):247–257, 1980.

[91] J. N. Yan, O. Schulte, M. Zhang, J. Wang, and R. Cheng. SCODED: statistical constraint oriented data error detection. In *SIGMOD*, pages 845–860, 2020.

[92] S. Yu, Z. Abraham, H. Wang, M. Shah, Y. Wei, and J. C. Príncipe. Concept drift detection and adaptation with hierarchical hypothesis testing. *J. Franklin Institute*, 356(5):3187–3215, 2019.

[93] Y. Zhang, Z. Guo, and T. Rekatsinas. A statistical perspective on discovering functional dependencies in noisy data. In *SIGMOD*, pages 861–876, 2020.

A SYSTEM PARAMETERS

Our technique for deriving (unnormalized) importance factor γ_k, for bounded-projection constraint on projection F_k, uses the mapping $\frac{1}{\log(2+\sigma(F_k(D)))}$. This mapping correctly translates our principles for quantifying violation by putting high weight on conformance constraints constructed from low-variance projections, and low weight on conformance constraints constructed from high-variance projections. While this mapping works extremely well across a large set of applications (including the ones shown in our experimental results), our quantitative semantics are not limited to any specific mapping. In fact, the function to compute importance factors for bounded-projections can be user-defined (but we do not require it from the user). Specifically, a user can plug in any custom function to derive the (unnormalized) importance factors. Furthermore, our technique to compute the bounds lb and ub can also be customized (but we do not require it from the user either). Depending on the application requirements, one can apply techniques used in machine learning literature (e.g., cross-validation) to tighten or loosen the conformance constraints by tuning these parameters. However, we found our technique—even without any cross-validation—for deriving these parameters to be very effective in most practical applications.

B PROOF OF LEMMA 11

Proof. Pick β_1, β_2 s.t. $\beta_1^2 + \beta_2^2 = 1$ and the following equation holds:

$$\texttt{sign}(\rho_{F_1,F_2})\beta_1\sigma(F_1(D)) + \beta_2\sigma(F_2(D)) = 0 \tag{4}$$

Let t be any tuple that is incongruous w.r.t. $\langle F_1, F_2\rangle$. Now, we compute how far t is from the mean of the projection F on D:

$$\begin{aligned}
|\Delta F(t)| &= |F(t) - \mu(F(D))| \\
&= |\beta_1 F_1(t) + \beta_2 F_2(t) - \mu(\beta_1 F_1(D) + \beta_2 F_2(D))| \\
&= |\beta_1 \Delta F_1(t) + \beta_2 \Delta F_2(t)| \\
&= |\beta_1 \Delta F_1(t)| + |\beta_2 \Delta F_2(t)|
\end{aligned}$$

The last step is correct only when $\beta_1\Delta F_1(t)$ and $\beta_2\Delta F_2(t)$ are of same sign. We prove this by cases:

(Case 1). $\rho_{F_1,F_2} \geq \frac{1}{2}$. In this case, β_1 and β_2 are of different signs due to Equation 4. Moreover, since t is incongruous w.r.t. $\langle F_1, F_2\rangle$, $\Delta F_1(t)$ and $\Delta F_2(t)$ are of different signs. Hence, $\beta_1\Delta F_1(t)$ and $\beta_2\Delta F_2(t)$ are of same sign.

(Case 2). $\rho_{F_1,F_2} \leq -\frac{1}{2}$. In this case, β_1 and β_2 have the same sign due to Equation 4. Moreover, since t is incongruous w.r.t. $\langle F_1, F_2\rangle$, $\Delta F_1(t)$ and $\Delta F_2(t)$ are of same sign. Hence, $\beta_1\Delta F_1(t)$ and $\beta_2\Delta F_2(t)$ are of same sign.

Next, we compute the variance of F on D:

$$\begin{aligned}\sigma(F(D))^2 &= \frac{1}{|D|}\sum_{t\in D}(\beta_1\Delta F_1(t)+\beta_2\Delta F_2(t))^2\\ &= \beta_1^2\sigma(F_1(D))^2+\beta_2^2\sigma(F_2(D))^2\\ &\qquad +2\beta_1\beta_2\rho_{F_1,F_2}\sigma(F_1(D))\sigma(F_2(D))\\ &= \beta_1^2\sigma(F_1(D))^2+\beta_1^2\sigma(F_1(D))^2-2\beta_1^2|\rho_{F_1,F_2}|\sigma(F_1(D))^2\\ &= 2\beta_1^2\sigma(F_1(D))^2(1-|\rho_{F_1,F_2}|)\end{aligned}$$

Hence, $\sigma(F(D)) = \sqrt{2(1-|\rho_{F_1,F_2}|)}|\beta_1|\sigma(F_1(D))$, which is also equal to $\sqrt{2(1-|\rho_{F_1,F_2}|)}|\beta_2|\sigma(F_2(D))$. Since $\sqrt{2(1-|\rho_{F_1,F_2}|)} \leq 1$, and since $|\beta_k| < 1$, we conclude that $\sigma(F(D)) < \sigma(F_k(D))$.

Now, we compute $\frac{|\Delta F(t)|}{\sigma(F(D))}$ next using the above derived facts about $|\Delta F(t)|$ and $\sigma(F(D))$.

$$\begin{aligned}\frac{|\Delta F(t)|}{\sigma(F(D))} &> \frac{|\beta_1\Delta F_1(t)|}{\sqrt{2(1-|\rho_{F_1,F_2}|)}|\beta_1|\sigma(F_1(D))}\\ &= \frac{|\Delta F_1(t)|}{\sqrt{2(1-|\rho_{F_1,F_2}|)}\sigma(F_1(D))} \qquad \geq \frac{|\Delta F_1(t)|}{\sigma(F_1(D))}\end{aligned}$$

The last step uses the fact that $|\rho_{F_1,F_2}| \geq \frac{1}{2}$. Similarly, we also get $\frac{|\Delta F(t)|}{\sigma(F(D))} > \frac{|\Delta F_2(t)|}{\sigma(F_2(D))}$. Hence, ϕ_F is stronger than both ϕ_{F_1} and ϕ_{F_2} on d, using Lemma 5. This completes the proof. □

C PROOF OF THEOREM 12

Proof. First, we use Lemma 11 on F_i, F_j to construct F. We initialize $I := \{i, j\}$. Next, we repeatedly do the following: We iterate over all F_k, where $k \notin I$, and check if $|\rho_{F,F_k}| \geq \frac{1}{2}$ for some k. If yes, we use Lemma 11 (on F and F_k) to update F to be the new projection returned by the lemma. We update $I := I \cup \{k\}$, and continue the iterations. If $|\rho_{F,F_k}| < \frac{1}{2}$ for all $k \notin I$, then we stop. The final F and index set I can easily be seen to satisfy the claims of the theorem. □

D PROOF OF THEOREM 13

We first provide some additional details regarding the statement of the theorem. Since standard deviation is not scale invariant, if there is no constraint on the norm of the linear projections, then it is possible to scale down the linear projections to make their standard deviations arbitrarily small. Therefore, claim (1) can not be proved for *any* linear projection, but only linear projections whose 2-norm is not too "small". Hence, we restate the theorem with some additional technical conditions.

Given a numerical dataset D, let $\mathcal{F} = \{F_1, F_2, \ldots, F_K\}$ be the set of linear projections returned by Algorithm 1. Let $\sigma^* = \min_k^K \sigma(F_k(D))$. WLOG, assume $\sigma^* = \sigma(F_1(D))$ where $F_1 = \vec{A}^T\vec{w}^*$. Assume that the attribute mean is zero for all attributes in D (call this Condition 1). Then,

(1) $\sigma^* \leq \sigma(F(D))$ for every possible linear projection F whose 2-norm is sufficiently large, i.e., we require $||\vec{w}|| \geq 1$ for $F = \vec{A}^T\vec{w}$. If we do not assume Condition 1, then the requirement changes to $||\vec{w}|| \geq ||\vec{w}^{*e}|| - \mu(D^T\vec{w})||$. Here $\vec{w}^{*e}$ is the vector constructed by augmenting a dimension to $\vec{w}^*$ to turn it to an eigenvector of ${D^e}^T D^e$ where $D^e = [\vec{1}; D]$.

(2) $\forall F_j, F_k \in \mathcal{F}$ s.t. $F_j \neq F_k$, $\rho_{F_j,F_k} = 0$. If we do not assume Condition 1, then the correlation coefficient is close to 0 for those F_j, F_k whose corresponding eigenvalues are much smaller than $|D|$.

Proof. The proof uses the following facts:

(Fact 1) If we add a constant c to each element of a set S of real values to get a new set S', then $\sigma(S) = \sigma(S')$.

(Fact 2) The Courant-Fischer min-max theorem [35] states that the vector $\vec{w}$ that minimizes $||M\vec{w}||/||\vec{w}||$ is the eigenvector of M^TM corresponding to the lowest eigenvalue (for any matrix M).

(Fact 3) Since $D'_N := [\vec{1}; D_N]$, by definition: $\sigma(D_N\vec{w}) = \frac{||D'_N\vec{w}'||}{\sqrt{|D|}}$, where $\vec{w}' = \begin{bmatrix}-\mu(D_N\vec{w})\\ \vec{w}\end{bmatrix}$

(Fact 4) By the definition of variance, $\sigma(S)^2 = ||S||^2 - \mu(S)^2$.

Let $F = \vec{A}^T\vec{w}$ be an arbitrary linear projection. Since D is numerical, $D = D_N$. Let D^e denote D'_N. (We use the superscript e to denote the augmented vector/matrix).

$$\begin{aligned}&\sigma(D^T\vec{w})^2\\ &= \sigma(D^T\vec{w} - \vec{1}\mu)^2 && \text{(Fact 1), } \mu = \mu(D^T\vec{w})\\ &= \sigma({D^e}^T\vec{w}^e)^2 && \text{where } \vec{w}^e = \begin{bmatrix}-\mu\\ \vec{w}\end{bmatrix}\\ &= \frac{||{D^e}^T\vec{w}^e||^2}{|D|} && \text{(Fact 3)}\\ &\geq \frac{||{D^e}^T\vec{w}^{*e}||^2\cdot||\vec{w}^e||^2}{|D|\cdot||\vec{w}^{*e}||^2} && \text{(Fact 2)}\\ &= (\sigma({D^e}^T\vec{w}^{*e})^2 + b^2)\cdot\frac{||\vec{w}^e||^2}{||\vec{w}^{*e}||^2} && \text{(Fact 4), } b = \mu({D^e}^T\vec{w}^{*e})\\ &= (\sigma(D^T\vec{w}^* + c)^2 + b^2)\cdot\frac{||\vec{w}^e||^2}{||\vec{w}^{*e}||^2} && \text{Expand } {D^e}^T\vec{w}^{*e}\\ &= (\sigma(D^T\vec{w}^*)^2 + b^2)\cdot\frac{||\vec{w}^e||^2}{||\vec{w}^{*e}||^2} && \text{(Fact 1)}\\ &= (\sigma^{*2} + b^2)\cdot\frac{||\vec{w}^e||^2}{||\vec{w}^{*e}||^2} && \text{definition of } \sigma^*\\ &\geq \sigma^{*2} && \text{by assumption } \frac{||\vec{w}^e||^2}{||\vec{w}^{*e}||^2} \geq 1\end{aligned}$$

For the last step, we use the technical condition that the norm of the extension of $\vec{w}$ (extended by augmenting the mean over $D\vec{w}$) is at least as large as the norm of extension of $\vec{w}^*$ (extended to make it an eigenvector of ${D^e}^T D^e$). When Condition 1 holds, $||\vec{w}^e||^2 = ||\vec{w}||^2$ (because $\mu(F(D))$ will be 0 and therefore, $\vec{w}^e = \begin{bmatrix}0\\ \vec{w}\end{bmatrix}$), and $||\vec{w}^{*e}||^2 = 1$ (for the same reason), and hence $\frac{||\vec{w}^e||^2}{||\vec{w}^{*e}||^2} \geq 1$.

For part (2) of the claim, let $F_i = \vec{A}^T\vec{w}_i$ for all i, where $\vec{w}_i$ are the coefficients of the linear projection F_i. Let $c_i = \mu(F_i(D))$.

(Fact 5) If Condition 1 holds, $\forall i\ c_i = 0$.

By construction of F_i's, we know that w_i can be extended to be an eigenvector $\begin{bmatrix} d_i \\ \vec{w}_i \end{bmatrix}$ of $D^{e\,T} D^e$ (with corresponding eigenvalue λ_i). In general,

(Fact 6) It is easy to work out that $d_i = \frac{-c_i}{1-\frac{\lambda_i}{|D|}}$.

Thus, we have:

$$
\begin{aligned}
&\rho_{F_j,F_k} \\
&= \frac{\sum_{t\in D} \Delta F_j(t)\Delta F_k(t)}{|D|\sigma(F_j(D))\sigma(F_k(D))} && \text{(definition of } \rho) \\
&= \frac{(D\vec{w}_j - c_j\vec{1})^T (D\vec{w}_k - c_k\vec{1})}{|D|\sigma(F_j(D))\sigma(F_k(D))} \\
&= \frac{(D^e \vec{w}_j^e)^T D^e \vec{w}_k^e}{|D|\sigma(F_j(D))\sigma(F_k(D))} && w_i^e = \begin{bmatrix} -c_i \\ \vec{w}_i \end{bmatrix} \\
&= \frac{\vec{w}_j^{e\,T} D^{e\,T} D^e \vec{w}_k^e}{|D|\sigma(F_j(D))\sigma(F_k(D))} \\
&= \frac{\vec{w}_j^{e\,T} \lambda_k \vec{w}_k^e}{|D|\sigma(F_j(D))\sigma(F_k(D))} && \text{(Fact 5,6), } D^{e\,T} D^e \vec{w}_k^e = \lambda_k \vec{w}_k^e \\
&= 0 && \text{(eigenvectors are orthogonal)}
\end{aligned}
$$

When Condition 1 does not hold, Fact 5 would not hold, but Fact 6 continues to hold, and hence by continuity, if $|\frac{\lambda_i}{|D|}|$ is close to 0, then d_i will be close to $-c_i$, and ρ_{F_j,F_k} will be close to 0.

□

E PROOF OF PROPOSITION 17

Proof. We show that the conformance constraint $\Phi := \forall f, g \in C : f(D) = g(D) \Rightarrow f(t) - g(t) = 0$, is the required conformance constraint over D to detect tuples that are unsafe with respect to C and $[D; Y]$.

First, we claim that Φ is a conformance constraint for D. For this, we need to prove that every tuple in D satisfies Φ. Consider any $t' \in D$. We need to prove that $f(t') = g(t')$ for all $f, g \in C$ s.t. $f(D) = g(D) = Y$. Since $t' \in D$, and since $f(D) = g(D)$, it follows that $f(t') = g(t')$. This shows that Φ is a conformance constraint for every tuple in D.

Next, we claim that Φ is *not satisfied* by exactly tuples that are unsafe w.r.t. C and $[D; Y]$. Consider any t' such that $\neg\Phi(t')$. By definition of Φ, it follows that there exist $f, g \in C$ s.t. $f(D) = g(D) = Y$, but $f(t') \neq g(t')$. This is equivalent to saying that t' is unsafe, by definition. □

F MOTIVATION FOR DISJUNCTIVE CONFORMANCE CONSTRAINTS

We now provide an example motivating the need for disjunctive conformance constraints.

Example 18. *PCA-based approach fails in cases where there exist different piecewise linear trends within the data. If we apply PCA to learn conformance constraints on the entire dataset of Fig. 9(a), it will learn two low-quality constraints, with very high variance. In contrast, partitioning the dataset into three partitions (Fig. 9(b)), and then learning constraints separately on each partition, will result in significant improvement of the learned constraints.*

G IMPLICATION FOR UNSAFE TUPLES

Here, we provide justification for our definition of unsafe tuples.

Proposition 19. *If $t \in \mathbf{Dom}^m$ is a unsafe tuple w.r.t. C and $[D; Y]$, then for any $f \in C$ s.t. $f(D) = Y$, there exists a $g \in C$ s.t. $g(D) = Y$ but $f(t) \neq g(t)$.*

Proof. By the definition of unsafe tuple, there exist $g, g' \in C$ s.t. $g(D) = g'(D) = Y$, but $g(t) \neq g'(t)$. Now, given a function $f \in C$ s.t. $f(D) = Y$, the value $f(t)$ can be either equal to $g(t)$ or $g'(t)$, but not both. WLOG, say $f(t) \neq g(t)$. Then, we have found a function g s.t. $g(D) = Y$ but $f(t) \neq g(t)$, which completes the proof. □

Note that even when we mark t as unsafe, it is possible that the learned model makes the correct prediction on that tuple. However, there is a good chance that it makes a different prediction. Hence, it is useful to be aware of unsafe tuples.

Conformance Constraints as Preconditions for Trusted Machine Learning

Let C denote a class of functions. Given a dataset D, suppose that a tuple t is unsafe. This means that there exist $f, g \in C$ s.t. $f(t) \neq g(t)$, but $f(D) = g(D)$. Now, consider the logical claim that $f(D) = g(D)$. Clearly, f is not identical to g since $f(t) \neq g(t)$. Therefore, there is a nontrivial "proof" (in some logic) of the fact that "for all tuples $t \in D : f(t) = g(t)$". This "proof" will use some properties of D, and let ϕ be the formula denoting these facts. If ϕ_D is the characteristic function for D, then the above argument can be written as,

$$\phi_D(\vec{A}) \Rightarrow \phi(\vec{A}), \quad \text{and} \quad \phi(\vec{A}) \Rightarrow f(\vec{A}) = g(\vec{A})$$

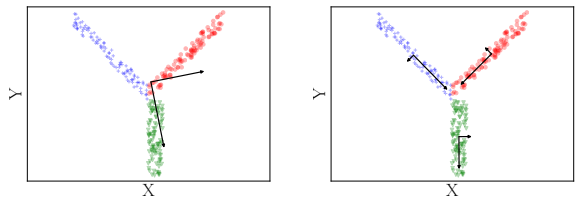
where $\Rightarrow$ denotes logical implication.

In words, ϕ is a conformance constraint for D and it serves as a *precondition* in the "correctness proof" that shows (a potentially machine-learned model) f is equal to (potentially a ground truth) g. If a tuple t fails to satisfy the precondition ϕ, then it is possible that the prediction of f on t will not match the ground truth $g(t)$.

Example 20. *Let $D = \{(0,1), (0,2), (0,3)\}$ be a dataset with two attributes A_1, A_2, and let the output Y be 1, 2, and 3, respectively. Let $C \subseteq ((\mathbb{R}\times\mathbb{R}) \mapsto \mathbb{R})$ be the class of linear functions over two variables A_1 and A_2. Consider a new tuple $(1, 4)$. It is unsafe since there exist two different functions, namely $f(A_1, A_2) = A_2$ and $g(A_1, A_2) = A_1 + A_2$, that agree with each other on D, but disagree on $(1, 4)$. In contrast, $(0, 4)$ is not unsafe because there is no function in C that maps D to Y, but produces an output different from 4. We apply Proposition 17 on the sets D, Y, and C. Here, C is the set of all linear functions given by $\{\alpha A_1 + A_2 \mid \alpha \in \mathbb{R}\}$. The conformance constraint Φ, whose negation characterizes the unsafe tuples w.r.t. C and $[D; Y]$, is $\forall\alpha_1, \alpha_2 : \alpha_1 A_1 + A_2 = \alpha_2 A_1 + A_2$, which is equivalent to $A_1 = 0$.*

Sufficient Check for Unsafe Tuples

In practice, finding conformance constraints that are necessary and sufficient for detecting if a tuple is unsafe is difficult. Hence, we focus on weaker constraints whose violation is sufficient, but not necessary, to classify a tuple as unsafe. We can use such constraints to get a procedure that has false negatives (fails to detect some unsafe tuples), but no false positives (never identifies a tuple as unsafe when it is not).



(a) PCA (b) Disjoint PCA

Figure 9: Learning PCA-based constraints globally results in low quality constraints when data satisfies strong local constraints.

Model Transformation using Equality Constraints. For certain conformance constraints, we can prove that a constraint violation by t implies that t is unsafe by showing that those constraints can transform a model f that works on D to a different model g that also works on D, but $f(t) \neq g(t)$. We claim that equality constraints (of the form $F(\vec{A}) = 0$) are useful in this regard. First, we make the point using the scenario from Example 20.

Example 21. *Consider the set C of functions, and the annotated dataset $[D; Y]$ from Example 15. The two functions f and g, where $f(A_1, A_2) = A_2$ and $g(A_1, A_2) = A_1 + A_2$, are equal when restricted to D; that is, $f(D) = g(D)$. What property of D suffices to prove $f(A_1, A_2) = g(A_1, A_2)$, i.e., $A_2 = A_1 + A_2$? It is $A_1 = 0$. Going the other way, if we have $A_1 = 0$, then $f(A_1, A_2) = A_2 = A_1 + A_2 = g(A_1, A_2)$. Therefore, we can use the equality constraint $A_1 = 0$ to transform the model f into the model g in such a way that the g continues to match the behavior of f on D. Thus, an equality constraint can be exploited to produce multiple different models starting from one given model. Moreover, if t violates the equality constraint, then it means that the models, f and g, would not agree on their prediction on t; for example, this happens for $t = (1, 4)$.*

Let $F(\vec{A}) = 0$ be an equality constraint for the dataset D. If a learned model f returns a real number, then it can be transformed into another model $f + F$, which will agree with f only on tuples t where $F(t) = 0$. Thus, in the presence of equality constraints, a learner can return f or its transformed version $f + F$ (if both models are in the class C). This condition is a "relevancy" condition that says that F is "relevant" for class C. If the model does not return a real, then we can still use equality constraints to modify the model under some assumptions that include "relevancy" of the constraint.

A Theorem for Sufficient Check for Unsafe Tuples. We first formalize the notions of *nontrivial* datasets—which are annotated datasets such that at least two output labels differ—and *relevant* constraints—which are constraints that can be used to transform models in a class to other models in the same class.

Nontrivial. An annotated dataset $[D; Y]$ is *nontrivial* if there exist i, j s.t. $y_i \neq y_j$.

Relevant. A constraint $F(\vec{A}) = 0$ is *relevant* to a class C of models if whenever $f \in C$, then $\lambda t : f(\texttt{ite}(\alpha F(t), t^c, t)) \in C$ for a constant tuple t^c and real number α. The if-then-else function $\texttt{ite}(r, t^c, t)$ returns t^c when $r = 1$, returns t when $r = 0$, and is free to return anything otherwise. If tuples admit addition, subtraction, and scaling, then one such if-then-else function is $t + r * (t^c - t)$.

We now state a sufficient condition for identifying unsafe tuples.

Theorem 22 (Sufficient Check for Unsafe Tuples). *Let $[D; Y] \subset \mathbf{Dom}^m \times \texttt{coDom}$ be an annotated dataset, C be a class of functions, and F be a projection on $\mathbf{Dom}^m$ s.t.*

A1. $F(\vec{A}) = 0$ is a strict constraint for D,
A2. $F(\vec{A}) = 0$ is relevant to C,
A3. $[D; Y]$ is nontrivial, and
A4. there exists $f \in C$ s.t. $f(D) = Y$.

For $t \in \mathbf{Dom}^m$, if $F(t) \neq 0$, then t is unsafe.

Proof. WLOG, let t_1, t_2 be the two tuples in D s.t. $y_1 \neq y_2$ (A3). Since $f(D) = Y$ (A4), it follows that $f(t_1) = y_1 \neq y_2 = f(t_2)$. Let t be a new tuple s.t. $F(t) \neq 0$. Clearly, $f(t)$ can not be equal to both y_1 and y_2. WLOG, suppose $f(t) \neq y_1$. Now, consider the function g defined by $\lambda\tau : f(\texttt{ite}(F(\tau), t_1, \tau))$. By (A2), we know that $g \in C$. Note that $g(D) = Y$ since for any tuple $t_i \in D$, $F(t_i) = 0$ (A1), and hence $g(t_i) = f(\texttt{ite}(0, t_1, t_i)) = f(t_i) = y_i$. Thus, we have two models, f and g, s.t. $f(D) = g(D) = Y$. To prove that t is a unsafe tuple, we have to show that $f(t) \neq g(t)$. Note that $g(t) = f(\texttt{ite}(F(t), t_1, t)) = f(t_1) = y_1$ (by definition of g). Since we already had $f(t) \neq y_1$, it follows that we have $f(t) \neq g(t)$. This completes the proof. □

We caution that our definition of unsafe is liberal: existence of even one pair of functions f, g—that differ on t, but agree on the training set D—is sufficient to classify t as unsafe. It ignores issues related to the probabilities of finding these models by a learning procedure. Our intended use of Theorem 22 is to guide the choice for the class of constraints, given the class C of models, so that we can use violation of a constraint in that class as an indication for caution. For most classes of models, linear arithmetic constraints are relevant. Our formal development has completely ignored that data (in machine learning applications) is noisy, and exact equality constraints are unlikely to exist. However, the development above can be extended to the noisy case by replacing exact equality is replaced by approximate equality. For example, when learning from dataset D and ground-truth f', we may not always learn a f that exactly matches f' on D, but is only close (in some metric) to f'. Similarly, equality constraints need not require $F(t) = 0$ for all $t \in D$, but only $F(t) \approx 0$ for some suitable definition of approximate equality. For ease of presentation, we have restricted ourselves to the simpler setting, which nevertheless brings out the salient points.

Example 23. *Consider the annotated dataset $[D; Y]$ and the class C, from Example 21. Consider the equality constraint $F(A_1, A_2) = 0$, where the projection F is defined as $F(A_1, A_2) = A_1$. Clearly, $F(D) = \{0\ 0\ 0\}$, and hence, $F(A_1, A_2) = 0$ is a constraint for D. The constraint is also relevant to the class of linear models C. Clearly, $[D; Y]$ is nontrivial, since $y_1 = 1 \neq 2 = y_2$. Also, there exists $f \in C$ (e.g., $f(A_1, A_2) = A_2$) s.t. $f(D) = Y$. Now, consider the tuple $t = (1, 4)$. Since $F(t) = 1 \neq 0$, Theorem 22 implies that t is unsafe.*

SQL Check Constraints

Due to the simplicity of the conformance language to express conformance constraints, they can be easily enforced as SQL check constraints to prevent insertion of unsafe tuples to a database.

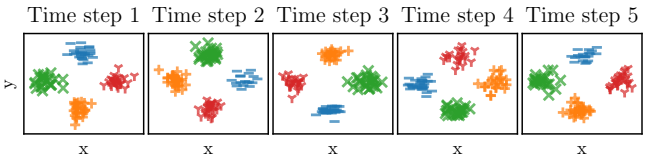


Figure 10: Snapshots over time for 4CR dataset with local drift. It reaches maximum drift from the initial distribution at time step 3 and goes back to the initial distribution at time step 5.

	lying	standing	sitting	walking	running
lying	0.05	0.41	0.57	0.68	0.78
standing	0.62	0.02	0.51	0.56	0.71
sitting	0.57	0.23	0.04	0.59	0.72
walking	0.21	0.01	0.06	0	0.25
running	0.12	0	0.03	0.02	0.01

Figure 11: Inter-activity constraint violation heat map. Mobile activities violate the constraints of the sedentary activities more.

H APPLICATIONS OF CONFORMANCE CONSTRAINTS

In database systems, conformance constraints can be used to detect change in data and query workloads, which can help in database tuning [47]. They have application in data cleaning (error detection and missing value imputation): the violation score serves as a measure of error, and missing values can be imputed by exploiting relationships among attributes that conformance constraints capture. Conformance constraints can detect outliers by exposing tuples that significantly violate them. Another interesting data management application is *data-diff* [76] for exploring differences between two datasets: our disjunctive constraints can explain how different partitions of two datasets vary.

In machine learning, conformance constraints can be used to suggest when to retrain a machine-learned model. Further, given a pool of machine-learned models and the corresponding training datasets, we can use conformance constraints to *synthesize* a new model for a new dataset. A simple way to achieve this is to pick the model such that constraints learned from its training data are minimally violated by the new dataset. Finally, identifying non-conforming tuples is analogous to *input validation* that performs sanity checks on an input before it is processed by an application.

I VISUALIZATION OF LOCAL DRIFT

When the dataset contains instances from multiple classes, the drift may be just local, and not global. Fig. 10 demonstrates a scenario for the 4CR dataset over the EVL benchmark. If we ignore the color/shape of the tuples, we will not observe any significant drift across different time steps.

J MORE DATA-DRIFT EXPERIMENTS

Inter-activity drift. Similar to inter-person constraint violation, we also compute inter-activity constraint violation over the HAR dataset (Fig. 11). Note the asymmetry of violation scores between activities, e.g., `running` is violating the constraints of `standing` much more than the other way around. A close observation reveals that, all mobile activities violate all sedentary activities more than the other way around. This is because, the mobile activities behave as a "safety envelope" for the sedentary activities. For example, while a person walks, they also stand (for a brief moment); but the opposite does not happen.

K EXPLAINING NON-CONFORMANCE

When a serving dataset is determined to be sufficiently deviated or drifted from the training set, the next step often is to characterize the difference. A common way of characterizing these differences is to perform a causality or responsibility analysis to determine which attributes are most responsible for the observed drift (non-conformance). We use the violation values produced by conformance constraints, along with well-established principles of causality, to quantify responsibility for non-conformance.

ExTuNe. We built a tool ExTuNe [24], on top of CCSynth, to compute the responsibility values as described next. Given a training dataset D and a non-conforming tuple $t \in \mathbf{Dom}^m$, we measure the *responsibility* of the i^{th} attribute A_i towards the non-conformance as follows: (1) We intervene on $t.A_i$ by altering its value to the mean of A_i over D to obtain the tuple $t^{(i)}$. (2) In $t^{(i)}$, we compute how many additional attributes need to be altered to obtain a tuple with no violation. If K additional attributes need to be altered, A_i has responsibility $\frac{1}{K+1}$. (3) This responsibility value for each tuple t can be averaged over the entire serving dataset to obtain an aggregate responsibility value for A_i. Intuitively, for each tuple, we are "fixing" the value of A_i with a "more typical" value, and checking how close (in terms of additional fixes required) this takes us to a conforming tuple. The larger the number of additional fixes required, the lower the responsibility of A_i.

Datasets. We use four datasets for this evaluation: (1) *Cardiovascular Disease* [1] is a real-world dataset that contains information about cardiovascular patients with attributes such as height, weight, cholesterol level, glucose level, systolic and diastolic blood pressures, etc. (2) *Mobile Prices* [4] is a real-world dataset that contains information about mobile phones with attributes such as ram, battery power, talk time, etc. (3) *House Prices* [3] is a real-world dataset that contains information about houses for sale with attributes such as basement area, number of bathrooms, year built, etc. (4) *LED* (Light Emitting Diode) [12] is a synthetic benchmark. The dataset has a digit attribute, ranging from 0 to 9, 7 binary attributes—each representing one of the 7 LEDs relevant to the digit attribute—and 17 irrelevant binary attributes. This dataset includes gradual concept drift every 25,000 rows.

Case studies. ExTuNe produces bar-charts of responsibility values as depicted in Fig. 12. Figures 12(a), 12(b), and 12(c) show the explanation results for Cardiovascular Disease, Mobile Price, and House Price datasets, respectively. For the cardiovascular disease dataset, the training and serving sets consist of data for patients without and with cardiovascular disease, respectively. For the House Price and Mobile Price datasets, the training and serving sets consist of houses and mobiles with prices below and above a certain threshold, respectively. As one can guess, we get many useful insights from

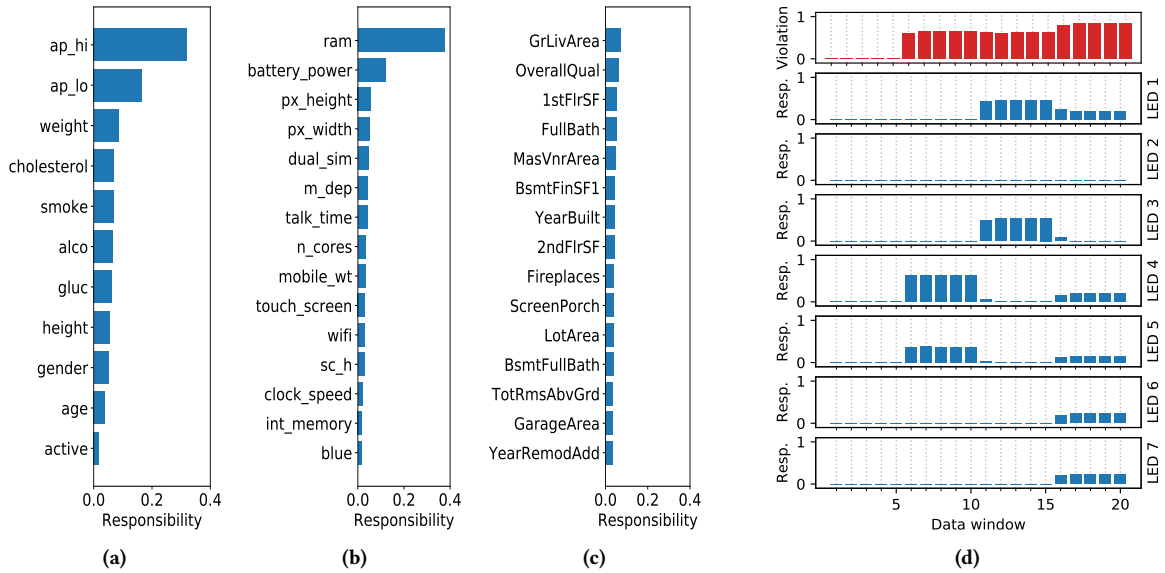


Figure 12: Responsibility assignment on attributes for drift on (a) Cardiovascular disease: trained on patients with no disease and served on patients with disease, (b) Mobile Prices: trained on cheap mobiles and served on expensive mobiles and (c) House Prices: trained on house with price <= 100K and served on house with price >= 300K. (d) Detection of drift on LED dataset. The dataset drifts every 5 windows (25,000 tuples). At each drift, a certain set of LEDs malfunction and take responsibility of the drift.

the non-conformance responsibility bar-charts such as: "abnormal (high or low) blood pressure is a key cause for non-conformance of patients with cardiovascular disease w.r.t. normal people", "RAM is a distinguishing factor between expensive and cheap mobiles", "the reason for houses being expensive depends holistically on several attributes".

Fig. 12(d) shows a similar result on the LED dataset. Instead of one serving set, we had 20 serving sets (the first set is also used as a training set to learn conformance constraints). We call each serving set a window where each window contains 5,000 tuples. This dataset introduces gradual concept drift every 25,000 rows (5 windows) by making a subset of LEDs malfunctioning. As one can clearly see, during the initial 5 windows, no drift is observed. In the next 5 windows, LED 4 and LED 5 starts malfunctioning; in the next 5 windows, LED 1 and LED 3 starts malfunctioning, and so on.

L CONTRAST WITH PRIOR ART

Simple conformance constraints vs. least square techniques. Note that the lowest variance principal component of $[\vec{1}; D_N]$ is related to the ordinary least square (OLS)—commonly known as linear regression—estimate for predicting $\vec{1}$ from D_N; but OLS minimizes error for the target attribute only. Our PCA-inspired approach is more similar to total least squares (TLS)—also known as orthogonal regression—that minimizes observational errors on all predictor attributes. However, TLS returns only the lowest-variance projection (Fig. 13(d)). In contrast, PCA offers multiple projections at once (Figs. 13(b), 13(c), and 13(d)) for a set of tuples (Fig. 13(a)), which range from low to high variance and have low mutual correlation (since they are orthogonal to each other). Intuitively, conformance constraints constructed from all projections returned by PCA capture various aspects of the data, as it forms a bounding hyper-box around the data tuples. However, to capture the relative importance of conformance constraints, we inversely weigh them according to the variances of their projections in the quantitative semantics.

Compound constraints vs. denial constraints. If we try to express the compound constraint ψ_2 of Example 3 using the notation from traditional denial constraints [17] (under closed-world semantics), where M always takes values from {"May", "June", "July"}, we get the following:

$$\Delta : \neg((M = \text{“May”}) \wedge \neg(-2 \leq AT - DT - DUR \leq 0))$$
$$\wedge \neg((M = \text{“June”}) \wedge \neg(0 \leq AT - DT - DUR \leq 5))$$
$$\wedge \neg((M = \text{“July”}) \wedge \neg(-5 \leq AT - DT - DUR \leq 0))$$

Note however that arithmetic expressions that specify linear combination of numerical attributes (highlighted fragment signifying a projection) are disallowed in denial constraints, which only allow raw attributes and constants within the constraints. Furthermore, existing techniques that compute denial constraints offer no mechanism to discover constraints involving such a composite attribute (projection). Under an open-world assumption, conformance constraints are more conservative—and therefore, more suitable for certain tasks such as TML—than denial constraints. For example, a new tuple with M = "August" will satisfy the above constraint Δ but not the compound conformance constraint ψ_2 of Example 3.

Data profiling. Conformance constraints, just like other constraint models, fall under the umbrella of data profiling using metadata [5]. There is extensive literature on data-profiling primitives that model

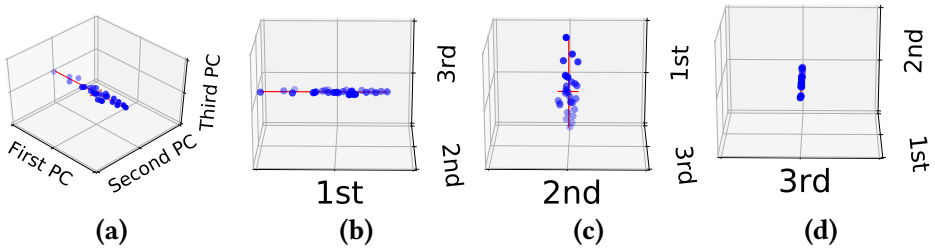


Figure 13: (a) 3D view of a set of tuples projected onto the space of principal components (PC). (b) The first PC gives the projection with highest standard deviation and thus constructs the weakest conformance constraint with a very broad range for its bounds. (c) The second PC gives a projection with moderate standard deviation and constructs a relatively stronger conformance constraint. (d) The third PC gives the projection with lowest standard deviation and constructs the strongest conformance constraint.

relationships among data attributes, such as unique column combinations [30], functional dependencies (FD) [59, 93] and their variants (metric [48], conditional [23], soft [38], approximate [36, 50], relaxed [16], etc.), differential dependencies [72], order dependencies [52, 77], inclusion dependencies [55, 60], denial constraints [13, 17, 53, 61], and statistical constraints [91]. However, none of them focus on learning approximate arithmetic relationships that involve multiple numerical attributes in a noisy setting, which is the focus of our work.

Soft FDs [38] model correlation and generalize traditional FDs by allowing uncertainty, but are limited in modeling relationships between only a pair of attributes. Metric FDs [48] allow small variations in the data, but the existing work focuses on verification only and not discovery of metric FDs. Some variants of FDs [16, 36, 48, 50] consider noisy setting, but they require the allowable noise parameters to be explicitly specified by the user. However, determining the right settings for these parameters is non-trivial. Most existing approaches treat constraint violation as Boolean, and do not measure the degree of violation. In contrast, we do not require any explicit noise parameter and provide a way to quantify the degree of violation of conformance constraints.

Conditional FDs [23] require the FDs to be satisfied conditionally (e.g., a FD may hold for US residents and a different FD for Europeans). Denial constraints (DC) are a universally-quantified first-order-logic formalism [17] and can adjust to noisy data, by adding predicates until the constraint becomes exact over the entire dataset. However, this can make DCs large, complex, and uninterpretable. While approximate denial constraints [61] exist, similar to approximate FD techniques, they also rely on the users to provide the error threshold.

Input validation. Our work here contributes to, while also building upon, work from machine learning, programming languages, and software engineering. In software engineering, input validation has been used to improve reliability [90]. For example, it is especially used in web applications where static and dynamic analysis of the code, that processes the input, is used to detect vulnerabilities [89]. For monitoring deployed systems, few prior works exploit constraints [40, 80]. To prevent unwanted outcomes, input validation techniques [15, 29] are used in software systems. However, such mechanisms are usually implemented by deterministic rules or constraints, which domain experts provide. In contrast, we learn conformance constraints in an unsupervised manner.

Trusted AI. The issue of trust, resilience, and interpretability of artificial intelligence (AI) systems has been a theme of increasing interest recently [39, 67, 86], particularly for high-stake and safety-critical data-driven AI systems [80, 87]. A standard way to decide whether to trust a classifier or not, is to use the classifier-produced confidence score. However, unlike classifiers, regressors lack a natural way to produce such confidence scores. To evaluate model performance, regression diagnostics check if the assumptions made by the model during training are still valid for the serving data. However, they require knowledge of the ground-truths for the serving data, which is often unavailable.

Data drift. Prior work on data drift, change detection, and covariate shift [7, 14, 18, 19, 21, 22, 33, 34, 37, 44, 46, 70, 73] relies on modeling data distribution, where change is detected when the data distribution changes. However, data distribution does not capture constraints, which is the primary focus of our work. Instead of detecting drift globally, only a handful of works model local concept-drift [82] or drift for imbalanced data [88]. Few data-drift detection mechanisms rely on availability of classification accuracy [11, 25, 26, 66] or classification "blindspots" [71]. Some of these works focus on adapting change in data, i.e., learning in an environment where change in data is expected [11, 27, 57, 75, 92]. Such adaptive techniques are useful to obtain better performance for specific tasks; however, their goal is orthogonal to ours.

Representation learning, outlier detection, and one-class classification. Few works [20, 31] , related to our conformance constraint-based approach, use autoencoder's [32, 65] input reconstruction error to determine if a new data point is out of distribution. Another mechanism [54] learns data *assertions* via autoencoders towards effective detection of invalid serving inputs. However, such an approach is task-specific and needs a specific system (e.g., a deep neural network) to begin with. Our approach is similar to outlier-detection approaches [49] that define outliers as the ones that deviate from a generating mechanism such as local correlations. We also share similarity with one-class-classification [79], where the training data contains tuples from only one class. In general, there is a clear gap between representation learning approaches (that models data likelihood) [6, 32, 43, 65] and the (constraint-oriented) data-profiling techniques to address the problem of trusted AI. Our aim is to bridge this gap by introducing conformance constraints that are more abstract, yet informative, descriptions of data, tailored towards characterizing trust in ML predictions.