Relative Keys: Putting Feature Explanation into Context

(Full Version)

ABSTRACT

Formal feature explanations strictly maintain perfect conformity but are intractable to compute, while heuristic methods are much faster but can lead to problematic explanations due to lack of conformity guarantees. We propose relative keys that have the best of both worlds. Relative keys associate feature explanations with a set of instances as context, and warrant perfect conformity over the context as formal explanations do, whilst being orders of magnitudes faster and working for complex blackbox models. Based on it, we develop CCE, a prototype that computes explanations with provably bounded conformity and succinctness, without accessing the models. We show that computing the most succinct relative keys is NP-complete and develop various algorithms for it under the batch and online models, all with provable guarantees. Using 9 real-life datasets and 7 state-of-the-art explanation methods, we demonstrate that CCE explains cases where existing methods cannot, and provides more succinct explanations with perfect conformity for cases they can; moreover, it is 2 orders of magnitude faster.

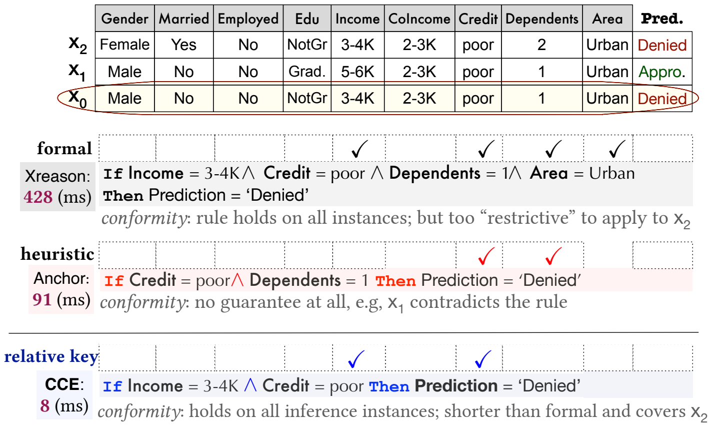
1 INTRODUCTION

There has been a significant success of machine learning (ML), in particular deep models on nontrivial tasks, *e.g.,* classification, natural language processing, and business intelligence. The success of these models comes at the cost of their complexity and blackbox inner workings, making it impossible for users to understand and reason about their behaviors. As they become increasingly central to high-stakes mission-critical tasks, there has been a growing interest in the problem of explaining the predictions of these models [29, 44, 45]. The European Union General Data Protection Regulation (GDPR) even explicitly requires "the right to explanation" [64, 88].

A popular and intuitive class of explanations is *feature explanation* [13, 48, 93, 96], Given an instance x, a feature explanation E of an ML model M for x is a parsimonious subset of x's features that in conjunction "determines" the prediction $M(\text{x})$ [51, 52, 61, 75, 76]. More concretely, E explains why M returns $M(\text{x})$ for x via a *rule-based semantics*: for any other instance x', if x' and x share the same values on features in E, then x' gets the same prediction from M as x does. Hence, it is also known as rule-based explanation [48, 76].

Conceptually, existing methods follow roughly the same 2-step routine: (i) generate a set of instances relevant to x and query model M to obtain their predictions; and (ii) derive an explanation E from the generated instances such that they conform to E under the rule-based explanation semantics. Obviously if E includes all features, it would be a valid explanation as all instances x' would conform to E. However, that does not reveal any information about what features really make M predict $M(\text{x})$. Hence, the objective is to minimize the succinctness of E, *i.e.,* number of features in E [13, 48, 76].

Enumerating all instances for step (i) is out of question due to the size of feature space. Step (ii) is also computationally expensive because of the search space. As a popular heuristic, Anchor [76], has been developed to sample instances close to x in the feature space for (i), and use a heuristic search for step (ii). It is shown effective



	Gender	Married	Employed	Edu	Income	CoIncome	Credit	Dependents	Area	Pred.
x_2	Female	Yes	No	NotGr	3-4K	2-3K	poor	2	Urban	Denied
x_1	Male	No	No	Grad.	5-6K	2-3K	poor	1	Urban	Appro.
x_0	Male	No	No	NotGr	3-4K	2-3K	poor	1	Urban	Denied
formal					✓		✓	✓	✓	
heuristic							✓	✓		
relative key					✓		✓			

Xreason: 428 (ms) `If` **Income** = 3-4K ∧ **Credit** = poor ∧ **Dependents** = 1∧ **Area** = Urban `Then` Prediction = 'Denied'
conformity: rule holds on all instances; but too "restrictive" to apply to x_2

Anchor: 91 (ms) `If` **Credit** = poor∧ **Dependents** = 1 `Then` Prediction = 'Denied'
conformity: no guarantee at all, e.g, x_1 contradicts the rule

CCE: 8 (ms) `If` **Income** = 3-4K ∧ **Credit** = poor `Then` **Prediction** = 'Denied'
conformity: holds on all inference instances; shorter than formal and covers x_2

Figure 1: Feature explanations of x_0 from the Loan dataset [4]

in trading the conformity [33] of explanation for efficiency [43, 76].

However, due to the lack of guarantees on conformity, explanations computed by heuristic methods may not always hold. As noted in [18, 47, 50, 62, 91], this can render explanations untrustworthy and problematic. To tackle this, there has been effort in developing *formal* explanation methods that rigorously implement the rule-based explanation semantics [21, 40, 48, 49, 62, 89].

Example 1: [Application Scenario] Banks often use third party ML-based assessment services [11, 12] to help with loan applications from their customers, who often request banks to provide explanations about their application outcomes. Assume that we are such an ML service provider for a bank that targets urban customers.

[Experiments] Following [48], we trained a XGBoost model [30] M using Loan [4] that contains 614 loan applications and their decisions. We then run Xreason [48] and Anchor [76], the state-of-the-art formal and heuristic feature explanation tools, respectively, to explain M's decision on urban Loan applications. Figure 1 (upper part) depicts their explanations for instance x_0 in Loan (simplified).

It took Xreason 428 ms to compute its explanation for x_0, which consists of `Income`, `Credit`, `Dependents` and `Area` of x_0. It guarantees that, for any instance x' in the feature space of M, it x' agrees with x_0 on these features, then $M(\text{x}')$ must also be "Denied". That is, it guarantees perfect conformity of the explanation: the rule-based semantics of its explanation holds for each and every instance.

On the other hand, Anchor is much faster, taking just 91 ms to find its explanation, which states that if an instance x' shares x_0's `Credit` and `Dependents` values, then $M(\text{x}')$ is consistent with $M(\text{x}_0)$, *i.e.,* "Denied". However, it does not guarantee that this is actually true. As a counter-example, x_1 in Loan (Fig. 1) agrees with x_0 on `Credit` and `Dependents` but has a different prediction by M. □

As shown in Example 1, formal methods strictly conform to the semantics of feature explanations, but are inefficient and often return lengthy explanations. While heuristic methods are faster, they do not guarantee the quality of explanations in the worst case and may contradict the semantics of their explanations. Hence, a natural but challenging question is *can we have the best of both worlds*?

We give a first attempt towards an affirmative answer.

Explanation in context. Our key proposition is that the rule-based semantics of the explanations does not need to be enforced against the entire feature space as *e.g.,* Xreason [48] does. Instead, only those instances that are relevant to the application need to conform to the explanation. For the case of Example 1, an application with `Area` = 'Rural' does not need to be examined against the explanation for conformity as it is irrelevant to the bank which targets urban applications only, although it is in the feature space of the model.

We refer to such relevant instances as the *context* of explanation, and for a given instance x, its explanation *w.r.t.* the context is called a *relative key*. Intuitively, relative keys are formal explanations *w.r.t.* the context. It reduces the search space of explaining while enforcing strict conformity of explanation semantics over instances in the context. A natural choice for the context is the inference set, *i.e.,* inference instances that are issued by and hence relevant to the application, and are also predicted by ML models during model serving.

Example 2: Continue with Example 1. As shown in the bottom part of Fig. 1, when using the inference set of Loan as the context, *i.e.,* urban Loan applications, the relative key for x_0 consists of `Income` and `Credit`, as succinct as that computed by Anchor and much shorter than that of Xreason. Unlike Anchor, it provides guarantees on the conformity of the explanation as Xreason does: for any Loan application x' that agrees with x_0 on `Income` and `Credit`, $M(x')$ must also be "Denied". As a result, using the relative key users can confidently decide that x_2 will be denied by M. This however cannot be deduced even from the formal explanation of Xreason despite of its conformity, since its explanation is too restrictive to cover x_2.

The relative key for x_0 was computed in 8 ms over Loan, as opposed to 428 ms and 91 ms by Xreason and Anchor, respectively. □

Client-centric explanation. Using relative keys with inference instances as context, we develop CCE, a framework for <u>C</u>lient-<u>C</u>entric feature <u>E</u>xplanation. It has the best of both worlds: (a) It is 226.4 times faster than Xreason, the state-of-the-art formal explanation method, and is even 43.0 times faster than heuristic Anchor. (b) It warrants that all inference instances strictly conform to explanations like formal methods do, but with more succinct explanations.

Better still, CCE does not need to query the ML model for explaining since the predictions of inference instances in the context are known for free during model serving. This gives the "right-to-explain" to users of arbitrary ML models, even those hosted in the cloud that provides no explanation interface, *e.g.,* bank in Example 1. As far as we know, all other feature explanations methods require accesses to the ML models due to step (i) of the explanation routine.

The foundation of CCE is the problem of computing relative keys, as a means of feature explanation, with an inference set as the context. Given an instance x to be explained, the problem is to compute the most succinct key for x relative to the context. We prove that this problem is intractable, *i.e.,* NP-COMPLETE. Nonetheless, we develop a PTIME approximation algorithm that outputs a relative key of size at most a logarithmic factor larger than the optimal.

Moreover, CCE goes beyond that by further allowing users to specify a ratio $\alpha \in (0, 1]$ such that it returns an *α-conformant* relative key that enforces conformity of explanations over an α-fraction of the context. CCE guarantees that its size is at most a logarithmic factor of the optimal one minus $\log 1/\alpha$. This gives users a flexible trade-off between the conformity and succinctness of explanations.

For cases where the client does not have the capacity to hold the inference set in its entirety as the context, or inference instances arrive online as a stream, CCE supports *explanation monitoring* that computes relative keys in an online manner, and maintains them as inference instances arrive. This also addresses practical needs in *e.g.,* behavior monitoring and adaptive ML [66, 71, 74, 94] where we want to monitor explanations in a dynamic setting. Again, we develop online algorithms for this with provable guarantees.

Effectiveness. Using 9 real-life datasets, we verify the effectiveness and benefit of relative keys by comparing CCE with 7 state-of-the-art explanation approaches. We find the following. (1) Relative key delivers the promise of maintaining rigorous conformity as formal explanations do on inference instances, in all test cases. (2) While warranting conformity, CCE is on average 2 (resp. 1) orders of magnitude faster than existing formal (resp. heuristic) methods. (3) Despite being formal and more efficient, relative keys are more accurate and succinct than traditional feature explanations. (4) As an application, CCE explains entity matching results with quality comparable to that of the state-of-the-art specialized entity matching explainer [95] over nearly half of the test cases, whilst being 4 orders of magnitude faster. In contrast, existing general-purpose explanation methods either does not work or is 10.4% and 2 orders of magnitude worse than CCE in quality and efficiency, respectively. (5) CCE can monitor dips in model prediction accuracy on-the-fly over inference set with noises or concept drift.

CCE also serves as a response to the recent call for "explainability for everyone", instead of just model owners [26, §4.11.4].

Contributions. In summary, we propose an approach to feature explanation from the user's perspective, by computing explanations with rigorous conformity like formal methods do over inference instances, while being orders of magnitude faster. Specifically:

- We propose and formalize the notion of relative keys, study its fundamental problem and settle the complexity (Section 3).
- We give an approximation algorithm for finding provably succinct α-conformant relative keys, for any $\alpha \in (0, 1]$ (Section 4).
- We develop online explanation monitoring algorithms that, for any $\alpha \in (0, 1]$, maintain α-conformant keys relative to a dynamic context with inference instances coming online (Section 5).
- We develop CCE, a client-centric feature explanation framework. Using 9 real-life datasets, we experimentally verify the benefit of relative keys and the effectiveness of CCE, by comparing with 7 state-of-the-art model explanation methods (Sections 6-7).

2 PRELIMINARY AND RELATED RESEARCH

ML basics. We consider models over discrete features (*a.k.a.* attributes). For a feature A, denote by $\text{dom}(A)$ its *domain*, *i.e.,* the set of all possible values that A draws from. For a list of features $A_1, \ldots, A_n$, we denote by $\mathcal{X}(A_1, \ldots, A_n)$ (or simply $\mathcal{X}$ when it is clear from the context) their *feature space*, defined as $\text{dom}(A_1)\times\cdots\times\text{dom}(A_n)$. An *instance* x over $A_1, \ldots, A_n$ is a tuple in $\mathcal{X}$. For any set $E \subseteq \{A_1, \ldots, A_n\}$, $x[E]$ is the projection of x on features in E.

An ML model M over $\mathcal{X}$ is a function that, given an instance $\mathrm{x} \in \mathcal{X}$, returns a prediction (*a.k.a.* label) $M(\mathrm{x}) \in \mathcal{Y}$ for x, where $\mathcal{Y}$ is referred to as the label space. The process of constructing M from pre-labeled data is known as model training. Using M to generate predictions for possibly unknown new instances is called model inference; such new instances are known as *inference instances*.

Feature explanations. Given a model M over $\mathcal{X}$, for any instance $\mathrm{x} \in \mathcal{X}$, a *feature explanation* of M for x is a subset of features of $\mathcal{X}$ that collectively determines the prediction $M(\mathrm{x})$ [61, 75, 76]. An instance $\mathrm{x}' \in \mathcal{X}$ *conforms to* explanation E for x if either $\mathrm{x}'[E] \neq \mathrm{x}[E]$ or $M(\mathrm{x}') = M(\mathrm{x})$. That is, if x' agrees with x on features in E, then they must share the same prediction by M when x' conforms to E. Observe that feature explanations use a rule-based semantics; hence they are also dubbed as rule-based explanations [56, 76, 80].

Conformity. For any feature explanation E of M for instance x, we say that E is *conformant* if for any other instance $\mathrm{x}' \in \mathcal{X}$, x' conforms to E. The *conformity* of a feature explanation method $\mathcal{A}$ is the probability of its explanations being conformant. In particular, if $\mathcal{A}$ has perfect conformity, *i.e.,* 100%, then it is called a *formal explanation method* and its explanations are also called formal explanations. Earlier studies [33, 41, 47, 50, 62] have shown that formal feature explanations can improve user trust in the ML model.

When E contains all the features of x, it is obviously a formal explanation for x. This is however not what we want. Instead, we fancy explanations that are as succinct as possible so that they apply to more instances and are easy to comprehend [48, 56, 76, 80].

Specifically, the *succinctness* of feature explanation E, denoted by $\mathrm{succinct}(E)$, is the number of features in E. We aim to compute explanations, ideally formal, but also as succinct as possible [25, 49]. This is challenging as formal explanations are often hard to find and tend to be less succinct [40, 80]. Indeed, deciding succinct formal feature explanation of a simple class of canonical decision trees is already Σ_2^p-hard [97], making it intractable to compute [21, 89].

Prior work. To ease the tension between conformity and efficiency of feature explanations, existing approaches adopt two directions: (a) heuristic methods [43, 76, 80] that trade conformity for better efficiency and succinctness, and (b) formal methods [48, 50] that stick to conformity by employing SAT solvers for computation.

(a) Heuristic. The dominating heuristic method is Anchor [76], which samples instances close to x by adding perturbations around it. It then uses the upper confidence bound heuristic [38] to derive an explanation E when optimizing its conformity over sampled instances. Anchor warrants no conformity guarantee for their explanations. Moreover, due to the complexity of heuristic, it is still quite slow if we want to avoid too low a conformity (more in Section 7.3).

(b) Formal explanations. The lack of guarantees, particularly on conformity, gives rise to situations where instances with the same feature explanation may have different predictions. This issue, as noted by recent studies [18, 47, 50, 62, 91], can render explanations untrustworthy and problematic. To tackle this, prime implicant [21, 40, 49, 62, 89], a formulation of formal explanation for decision trees, has been proposed and analyzed. The state of the art method for computing prime implicant is Xreason [48], which employs a SAT solver to enforce conformity. In addition to the high complexity, a prime implicant for x could still have unbounded succinctness.

All formal explanation methods, including Xreason, assume a simple white-box model and the knowledge of its structure, *e.g.,* (an ensemble of) decision trees [48, 89]. While heuristic methods also work for black-box models, they often use the distribution of training instances to assist in explanation [75, 76]. In addition, both types of methods assume the ownership of M and demand accesses to M for explanation. In contrast, CCE needs none of these. Moreover, with relative keys CCE is able to guarantee perfect conformity over inference set while being orders of magnitude faster than formal methods, even faster than heuristic method. In addition to conformity, it also provides provable bounds on the succinctness of its explanations that neither formal nor heuristic methods can offer.

Broader context of explanations. We next discuss additional related work from the wider perspective of explainability.

Global vs. local model explanations. Model explanations can be categorized into two classes: global explanations [29, 39, 56] and local explanations [75, 76, 80]. Local explanations interpret the prediction of an individual instance x by a model M. Feature explanations introduced earlier is one popular type of local explanations in the form of a parsimonious subset of feature values of x.

In contrast, global explanations interpret model M as a whole, without tailoring to an individual instance. They either (a) approximate a complex M with a simpler surrogate (*e.g.,* linear or tree) model [30], or (b) use patterns, *i.e.,* logic rules, over the feature space [15, 67] or the prediction datasets [39, 56] to capture the overall behavior of M and summarize its prediction outcomes [56, 67].

Local explanations have gained more attention as they are tailored and relevant to the instance to explain [29, 102], while global explanations are found more susceptible to over-generalization and redundancy, and often incur high computational overhead [19, 67]. In particular, due to sheer size of the search space, pattern-level explanations are necessarily heuristic [29, 39, 56]; as a result, they cannot always guarantee to explain a given instance.

More local explanations. We discuss other local explanations.

(1) Feature importance based explanations, *e.g.,* LIME [75] and SHAP [61], are another popular class of local explanations. They explain the prediction $M(\mathrm{x})$ of a given instance x by computing a coefficient (*a.k.a.* importance score) for each feature of x. They are orthogonal to feature explanations and provide more information. However, it has been argued that the quality of the coefficients is hard to evaluate [35, 79] and a large vector of importance scores can be difficult to perceive [67] when x has, *e.g.,* hundreds of features.

(2) Instance-based explanations [16, 67, 68, 73] are another orthogonal but well-received class of local explanations. They select a set of instances from the feature space $\mathcal{X}$ to collectively provide insight into why M returns prediction $M(\mathrm{x})$ for a given instance x, via counterfactuals [16, 68] or prototype [67] explanations. Instances are picked such that each of them is similar to x but differs in a controlled manner; by comparing them with x, we hope to understand the behavior of M for x and its neighbors in $\mathcal{X}$.

Domain-specific explanations. There has also been a host of work on domain-specific explanations, particularly in database community.

(a) Query explanations. Explanations for query results have been

	Gender	Income	Credit	Dependent	Prediction
x_0	Male	3-4K	poor	1	Denied
x_1	Male	5-6K	poor	1	Approved
x_2	Female	3-4K	poor	2	Denied
x_3	Male	3-4K	poor	1	Denied
x_4	Male	1-2K	poor	1	Denied
x_5	Male	3-4K	good	0	Approved
x_6	Male	3-4K	good	1	Approved

Manual Step $\Rightarrow$

Liquidity	Decision
high	Approved
low	Approved
low	Denied
low	Denied
high	Denied
high	Approved
high	Approved

Figure 2: An example context

extensively studied in the form of *e.g.,* provenance [27, 28, 42] and causality [63, 85] for non-aggregate queries, and their counterparts [65, 77, 78, 101] for aggregate queries. They are also closely related to the classic view-update [83, 84] and database repairs [82] problems. Conceptually, they work by computing an "intervention", *i.e.,* addition or removal of a set of tuples that can change the query answers [85], and using *e.g.,* predicates or patterns [65] to succinctly capture it, in a similar spirit of counterfactuals for ML models. More related is the variable-based explanations [86] that identify variables of queries that contribute most to the bias in query results. They work by analyzing query structures (*i.e.,* whitebox) while feature explanations also target complex blackbox ML models, *e.g.,* DNN.

(b) Entity-matching explanations. Closer is the study of explanations [23, 95] for ML-based entity-matching [24, 58, 70]. They adopt feature importance explanations, but improve them by taking into account the semantics of entity matches [95]. Hence, they have better quality than general ML explanation methods. Despite this, as shown in Section 7, CCE achieves comparable quality (*e.g.,* faithfulness [14, 20]) to state-of-the-art specialized entity-matching explainer [95] in nearly half of the cases and is orders of magnitude faster; in contrast, Anchor is consistently significantly worse in quality and Xreason cannot work. CCE also offers strict conformity.

As stated earlier, we focus on feature explanations in this work.

3 RELATIVE KEYS

We first present the notion of relative keys (Section 3.1). We then formulate the main problem and settle its complexity (Section 3.2).

3.1 Putting Feature Explanation into Context

Relative keys. Consider an ML model M over feature space $\mathcal{X}(A_1, \ldots, A_n)$ and label space $\mathcal{Y}$. Let $\mathbf{I}$ be a collection of instances in $\mathcal{X}$ and x is an instance in $\mathbf{I}$. A set $E \subseteq \{A_1, \ldots, A_n\}$ is an *key of* M *for* x *relative to* $\mathbf{I}$ if for any instance $\mathrm{x}' \in \mathbf{I}$, if $\mathrm{x}'[E] = \mathrm{x}[E]$ then $M(\mathrm{x}') = M(\mathrm{x})$; in this case $\mathbf{I}$ is referred to as a *context* of M. For convenience, E is also simply called an *relative key* for x.

Example 3: Consider a sample context $\mathbf{I}_0$ in Fig. 2, using simplified Loan schema in Example 1. $\mathbf{I}_0$ contains 7 loan application instances and their predictions by a model M. Then the key for x_0 relative to $\mathbf{I}_0$ is $E_1 = \{\texttt{Income}, \texttt{Credit}\}$, as every other instance of $\mathbf{I}_0$ with the same `Credit` and `Income` values as x_0 is also denied by M. □

Intuitively, relative keys capture formal feature explanations in a context $\mathbf{I}$: in contrast to traditional formal feature explanations that impose conformity on the entire feature space $\mathcal{X}$, relative keys enforce the semantics of formal feature explanations on context $\mathbf{I}$ that consists of, *e.g.,* only those instances of interest or relevant.

By varying $\mathbf{I}$, we can use relative keys to fine-tune formal explanations for applications. In particular, $\mathbf{I}$ referred to as a *inference context* when it collects inference instances. Since inference instances are often those relevant to the users, keys relative to an inference context can avoid the over-generalization of formal explanations while still upholding perfect conformity over instances that matter.

Generalization. Like traditional formal explanations, although relative keys assure perfect conformity *w.r.t.* $\mathbf{I}$, they can sometimes be too restrictive or not sufficiently succinct to interpret. However, instead of settling with heuristics, the context of relative keys enables us to relax the notion with rigorously controlled quality.

α-conformant relative keys. Consider a context $\mathbf{I}$ over feature space $\mathcal{X}$, instance $\mathrm{x} \in \mathbf{I}$ and ML model M. Let α be a ratio in $(0, 1]$. A set E_α of features of $\mathcal{X}$ is an *α-conformant key of M for* x *relative to* $\mathbf{I}$ if E_α is a key for x relative to at least an α-fraction of $\mathbf{I}$. That is, there exits $\mathbf{I}^\alpha \subseteq \mathbf{I}$ such that (a) $|\mathbf{I}^\alpha| \geq \alpha|\mathbf{I}|$ and (b) for any instance $\mathrm{x}' \in \mathbf{I}^\alpha$, if $\mathrm{x}'[E_\alpha] = \mathrm{x}[E_\alpha]$ then $M(\mathrm{x}') = M(\mathrm{x})$. Here α is referred to as a *conformity bound*, lower bounding the conformity of E_α.

Intuitively, when $\alpha = 1$, E_α reduces to the relative key for x, with perfect conformity. By varying α, one can trade the conformity of E_α as a feature explanation for better succinctness.

Example 4: Continue with Example 3. A $\frac{6}{7}$-conformant key E_3 for instance x_0 relative to $\mathbf{I}_0$ in Fig. 2 is {Credit}, smaller than E_1. □

Benefits. Relative keys bring several immediate benefits.

(a) Model-agnostic succinct formal explanations. Unlike existing formal explanations that assume white-box models, *e.g.,* decision trees, relative keys are defined *w.r.t.* an inference context $\mathbf{I}$, of which the predictions are known during model inference. Hence, they are model-agnostic and work for complex black-box models.

(b) Observing feature associations. Feature associations, such as dependencies, correlations or integrity constraints from application semantics, are commonly observed [31, 34, 41, 90]. Hence, instances that are relevant, *e.g.,* inference instances, often constitute only a fraction of the feature space. By focusing on inference context $\mathbf{I}$, relative keys can observe feature associations from the applications and provide more concise explanations than existing formal explanations that uniformly assert conformity over the entire feature space.

(c) Client-centric inference-time explanation. Since relative keys only reason about the inference context $\mathbf{I}$, independent of the model and training set, they can be computed at inference time. Moreover, $\mathbf{I}$ is readily available at the client side, by collecting interactions with possibly remote ML model inference services. This makes relative keys an ideal fit for client-centric feature explanation, without the ownership of or any explainability support from the ML models.

(d) Supporting hybrid ML workflow. In practice, predictions are often the decision-making of a hybrid workflow consisting of both ML models and rule-based or human-in-the-loop steps[1, 5]. To explain these decisions, it is necessary to go beyond interpreting the ML models, as the rules and manual steps are not part of the models' logic. By reasoning about the results of the entire workflow via $\mathbf{I}$, relative keys serve as a holistic explanation for decision-making.

Example 5: Continue with Example 3. Assume that the bank incorporates a manual step that makes final decisions based on predictions from M and other real-time factors, *e.g.,* Liquidity of the bank, and changes decision for instance x_0 from "Denied" to "Approved" as also shown in Fig. 2 (right). Then E_4 = {Income, Liquidity} is a key for x_0 relative to $\mathbf{I}_1$ ($\mathbf{I}_0$ extended with Liquidity), serving as a holistic explanation of the combine workflow for x_0. This is not possible if we explain by interpreting the ML model alone. □

3.2 Problem and Complexity

Problem statement. Key to feature explanation is to compute succinct and trustworthy explanations. We formulate this for relative keys, referred to as the *minimum relative key* problem (Mrkp):

Input	a context $\mathbf{I}$ of instances over feature space $\mathcal{X}(A_1, \ldots, A_n)$ and their predictions made by a (blackbox) ML model M, an instance $x \in \mathbf{I}$, and conformity bound $\alpha \in (0, 1]$.
Output	an α-conformant key E of M for x relative to $\mathbf{I}$.
Objective	minimize succinct(E).

Intuitively, Mrkp is to compute the most succinct α-conformant relative keys for any targeted conformity bound α and instance x.

Complexity. Consider the decision problem of Mrkp.

Theorem 1: *Problem* Mrkp *is* NP*-complete, even* $\alpha = 1$. □

Proof sketch: Mrkp is in NP as one can check whether a set E of features is an α-conformant relative key for x relative to $\mathbf{I}$ of bounded size in polynomial time. We show that Mrkp is NP-hard, even if $\alpha = 1$, by reduction from the minimum set cover (Msc) problem, which is known NP-complete [37]. Given an instance of Msc, we construct a context $\mathbf{I}$ and an instance x such that Msc has a k-cover if and only if there exists a k-minimum key for x relative to $\mathbf{I}$. □

Approximability. The intractability of Theorem 1 naturally motivates us to study approximate solutions, *i.e.,* efficient polynomial-time (Ptime) algorithms that compute α-conformant relative keys close to the most succinct ones. This suggests the following notion.

p-boundedness. An α-conformant key E_α for x relative to $\mathbf{I}$ is *p-bounded* if for *any* other α-conformant key E'_α for x relative to $\mathbf{I}$, we have that succinct$(E_\alpha) \leq p \cdot$ succinct(E'_α). Intuitively, p measures how far the number of features in an α-conformant relative key E_α is from the optimal (*i.e.,* most succinct) α-conformant relative key.

Denote by $|\mathbf{I}|$ the number of instances in $\mathbf{I}$. Then we show below:

Theorem 2: *(1) There exists* no Ptime *algorithm for* Mrkp *that returns a* $(1 - o(1)) \cdot \ln \alpha|\mathbf{I}|$*-bounded relative key, even if* $\alpha = 1$.

(2) There exists a Ptime *algorithm for* Mrkp *that, given any* $\alpha \in (0, 1]$*, computes an* α*-conformant* $\ln(\alpha|\mathbf{I}|)$*-bounded relative key.* □

Proof sketch: (1) We show that the NP-hardness reduction in the proof of Theorem 1 is also an approximation-preserving reduction [22]. Since Msc cannot be approximated within an approximation factor of $(1 - o(1)) \cdot \ln m$ in polynomial time [22], so does Mrkp.

(2) We give a constructive proof in Section 4. □

4 PRECISE AND SUCCINCT RELATIVE KEYS

As a proof of Theorem 2(2), we develop an algorithm for Mrkp.

Algorithm. Given an inference context $\mathbf{I}$ of an ML model M, an instance x_0 of $\mathbf{I}$, and conformity bound $\alpha \in (0, 1]$, we develop an algorithm, denoted by SRK, that returns an α-conformant p-bounded key for x_0 relative to $\mathbf{I}$ in polynomial time, where p is a function of α.

ALGORITHM 1: Algorithm SRK

```
Input: context $\mathbf{I}$, instance $x_0 = (a_1, \ldots, a_n) \in \mathbf{I}$, prediction $M(x)$
       for each $x \in \mathbf{I}$, conformity bound $\alpha \in (0, 1]$.
Output: an $\alpha$-conformant key $E_\alpha$ for $x$ relative to $\mathbf{I}$.
1 $\mathbf{A} \leftarrow \{1, \ldots, n\}$ ;        // $\mathbf{A}$: features of $\mathcal{X}$, i.e., $i$ corresponds to feature $A_i$
2 $i_0 \leftarrow \arg\min_{i \in \mathbf{A}} |\mathbf{I}[A_i = a_i] \setminus \mathbf{I}_{M(x_0)}|$ ;
3 $E \leftarrow E \cup \{i_0\}$; $\mathbf{A} \leftarrow \mathbf{A} \setminus \{i_0\}$;
4 while $|\bigcap_{j \in E} \mathbf{I}[A_j = a_j] \cap \mathbf{I}^c_{M(x_0)}| > (1 - \alpha) \cdot |\mathbf{I}|$ do
5     $i \leftarrow \arg\min_{i \in \mathbf{A}} |\bigcap_{j \in E} \mathbf{I}[A_j = a_j] \cap \mathbf{I}[A_i = a_i] \setminus \mathbf{I}_{M(x_0)}|$;
6     $E \leftarrow E \cup \{i\}$; $\mathbf{A} \leftarrow \mathbf{A} \setminus \{i\}$;
7 return $E$;
```

We use the following notations. Denote by $\mathbf{I}[A_i = c]$ the set of instances in $\mathbf{I}$ with $A_i = c$. Let $\mathbf{I}_{M(x_0)}$ be the set of all instances in $\mathbf{I}$ that share the same prediction with x_0, *i.e.,* $\mathbf{I}_{M(x_0)} = \{x \in \mathbf{I} \mid M(x) = M(x_0)\}$. Accordingly, denote by $\mathbf{I}[A_i \neq c]$ the set of instances in $\mathbf{I}$ with $A_i \neq c$. Let $\mathbf{I}^c_{M(x_0)}$ be the complement of $\mathbf{I}_{M(x_0)}$, *i.e.,* $\mathbf{I} \setminus \mathbf{I}_{M(x_0)}$. Let x_0 be $(a_1, \ldots, a_n)$ and its prediction by M be $M(x_0)$.

The idea of SRK is simple: it picks features of x_0 one by one, each time with the one that minimizes the number of instances of $\mathbf{I}$ that agree with x_0 on the selected features while having a different prediction. More specifically, as shown in Algorithm 1, SRK first identifies an initial feature (lines 1-3) and then iteratively picks features of x_0 one by one (lines 4-6). Each time it picks A_i such that

$$|\bigcap_{A_j \in E} \mathbf{I}[A_j = a_j] \cap \mathbf{I}[A_i = a_i] \setminus \mathbf{I}_{M(x_0)}|$$

is the minimum among all the A_i features that are not yet in E (line 5). The iteration completes and E is returned when

$$|\bigcap_{A_j \in E} \mathbf{I}[A_j = c_j] \cap \mathbf{I}^c_{M(x_0)}| \leq (1 - \alpha) \cdot |\mathbf{I}|.$$

Intuitively, the left side of the inequality represents the number of instances in $\mathbf{I}$ that agree with x_0 on features in E but have a different prediction. The right side denotes the tolerable number of instances that do not conform to the rule-based semantics of the relative keys according to the targeted conformity bound α.

Example 6: Continue with Example 3. To compute relative key for x_0, SRK first picks Credit for E, as only one instance (*i.e.,* x_1) in $\mathbf{I}_0$ that matches x_0 in Credit but predicts differently. It then adds Income to E via lines 5-6 as there is no other instance that agrees with x_0 on both Credit and Income but predicts differently. Then, condition of line 4 holds and {Credit, Income} is returned as the key.

When $\alpha = \frac{6}{7}$, line 4 holds when E has Credit only. Hence {Credit} is returned as a $\frac{6}{7}$-conformant key for x_0 relative to $\mathbf{I}_0$. □

Analysis. SRK can be implemented in $O(n^2|\mathbf{I}|)$-time, where n is the total number of features of $\mathbf{I}$ and $|\mathbf{I}|$ is the number of instances in $\mathbf{I}$. The algorithm, although quite simple, warrants the following.

Lemma 3: *Given any conformity bound* $\alpha \in (0, 1]$, SRK *always returns an* α*-conformant* $ln(\alpha|\mathbf{I}|)$*-bounded key for* x *relative to* $\mathbf{I}$. □

By Lemma 3, algorithm SRK is indeed a constructive proof of

ALGORITHM 2: Algorithm OSRK

```
Input: ML model $M$, instance $x_0$, conformity bound $\alpha \in (0,1]$,
       and context $\mathbf{I}$ that receives a stream of instances $x_1, x_2$,
       ...arriving online (assume w.l.o.g. $\mathbf{I}$ is empty initially).
Output: Upon the arrival of $x_t$ ($t \in [1,+\infty)$), an $\alpha$-conformant
        key $E_t$ for $x_0$ relative to $\mathbf{I} = \{x_1, \ldots, x_t\}$ (assume $E_0 = \emptyset$).
Upon the arrival of instance $x_t$:
1  $\mathbf{I} \leftarrow \mathbf{I} \cup \{x_t\}$; $E_t \leftarrow E_{t-1}$;   // ensure explanation coherence
2  if $M(x_t) = M(x_0)$ then return $E_t$;   // done; move onto next instance
3  if $M(x_t) \neq M(x_0)$ and $E_t = \emptyset$ then
4      foreach $i \in \mathbf{A}$ do  // $\mathbf{A} = \{1, \ldots, n\}$; $i$ corresponds to feature $A_i$ in $\mathcal{X}$
5          $w_i = 2^{-k}$ ;   // $k$ is the maximum integer for which $2^{-k} < 1/n$
6          add $i$ to $E_t$ with probability $w_i$;
7  $S_t = \{i \in \mathbf{A} \setminus E_t | x_t[A_i] \neq x_0[A_i]\}$;
8  while $|\bigcap_{j \in E_t} \mathbf{I}[A_j = x_0[A_j]] \cap \mathbf{I}^c_{M(x_0)}| > (1-\alpha) \cdot |\mathbf{I}|$ do
9      $\mu_t = \sum_{i \in S_t} w_i$;
10     if $\mu_t > \log p_t$ then   // $x_t$ is the $p_t$-th online instance where $M(x_t) \neq M(x_0)$
11         pick an arbitrary $i$ from $S_t$ and add it to $E_t$; break;
12     else   // $\mu_t \leq \log p_t$, performs the below weight augmentation
13         foreach $i \in S_t$ do
14             if $w_i < 1$ then $w_i = 2w_i$ ;
15             add $i$ to $E_t$ with probability $w_i$;
16 return $E_t$;   // done; move onto the next incoming instance $x_{t+1}$
```

Theorem 2(2). Below we sketch a proof of Lemma 3.

Proof: Denote by OPT the optimal algorithm that finds relative keys with minimum succinctness. Since we consider α-conformant relative keys, the conformity condition that E determines predictions only needs to hold on an α-fraction of $\mathbf{I}$. Let G_{SRK} (resp. G_{OPT}) be the number of instances that satisfy the dependency condition of relative keys over the features picked by SRK (resp. OPT). Assume *w.l.o.g.* $G_{\text{OPT}} = \lceil \alpha |\mathbf{I}| \rceil$. Consider the least value of k_{SRK}, denoted by r, such that $G_{\text{SRK}} \geq G_{\text{OPT}} - c$ for a positive constant c. By proving

$$G_{\text{SRK}} \geq \left(1 - (1 - \frac{1}{k_{\text{OPT}}})^{k_{\text{SRK}}}\right) G_{\text{OPT}}, \text{ we have}$$

$$\left(1 - (1 - \frac{1}{k_{\text{OPT}}})^{r}\right) G_{\text{OPT}} = G_{\text{OPT}} - c \Rightarrow \frac{r}{k_{\text{OPT}}} \leq \ln G_{\text{OPT}} - \ln c.$$

Since the r features serve as the relative keys over $G_{\text{OPT}} - c$ instances so far and each feature covers at least one new instance. Thus, $k_{\text{SRK}} \leq r + c$. Based on these, we have

$$\frac{k_{\text{SRK}}}{k_{\text{OPT}}} \leq \frac{r+c}{k_{\text{OPT}}} \leq \ln G_{\text{OPT}} = \ln \lceil \alpha |\mathbf{I}| \rceil. \qquad \square$$

Lemma 3 assures us that SRK can always efficiently return α-conformant explanations with succinctness that is within a logarithmic factor of the optimal ones, for any targeted conformity α.

5 RELATIVE KEYS WITH DYNAMIC CONTEXT

We next study a variant of Mrkp, where context $\mathbf{I}$ is a stream of incoming instances and the task is to update explanations on-the-fly. Below we first formulate the problem (Section 5.1). We then develop online algorithms for it with guarantees (Sections 5.2-5.3).

5.1 Monitoring Keys with a Dynamic Context

So far, we have seen how relative keys serve client-centric feature explanation and how we compute succinct and precise relative keys with a static context $\mathbf{I}$. We now consider its dynamic version: how can we compute and update relative keys when $\mathbf{I}$ changes over time, *i.e.,* collecting incoming new inference instances predicted online?

This is also motivated by recent calls for explanation monitoring methods to update explanations in response to new observations for applications in credit-risk evaluation [80] and FinTech [8], and also from the dynamic explainability community [10, 46, 69]. As will be also shown shortly in Section 7, as a by-product monitoring relative keys also helps us detect performance drifts of blackbox ML models or remote ML services that we are not in control of.

More specifically, we study the following variant of Mrkp, referred to as the *online relative key monitoring* (Orkm) problem:

Input	a (blackbox) ML model M, instance x_0, conformity bound $\alpha \in (0,1]$, a context $\mathbf{I}$ that collects instances $x_1, x_2, \ldots$ arriving and being predicted by M online;
Output	an α-conformant key E_i of M for x_0 relative to $\mathbf{I}$, upon the arrival of each $x_i (i = 1, 2, \ldots)$ in $\mathbf{I}$.
Objective	minimize succinct(E_i), for each $i = 1, 2, \ldots$
Constraint	the relative keys are *coherent*: for any i, $E_i \subseteq E_{i+1}$.

Intuitively, Orkm is an online variant of problem Mrkp, where instances in $\mathbf{I}$ is revealed one by one on-the-fly and we need to return a succinct α-conformant key relative to the updated $\mathbf{I}$ upon the revelation of each instance when $\mathbf{I}$ grows. Moreover, Orkm complies with the *explanation coherence* of dynamic explainability [46, 55, 81], as users tend to trust explanations that resemble previous versions. The coherence property prevents explanations from jumping randomly from one set of features to an unrelated new set of features in successive rounds, and this has proven critical in maintaining user trust in ML models [59, 66, 92, 98, 100]. Note that, if E_i is a minimal key for x_0 relative to $\mathbf{I}$, then removing any feature from E_i will make E_i not a key for x_0 relative to $\mathbf{I} \cup \{x_{i+1}\}$, for any new instance x_{i+1}. Hence, we only grow the relative keys as $\mathbf{I}$ expands, *i.e.,* $E_i \subseteq E_{i+1}$ implements the explanation coherence.

While critically important, the problem is inherently challenging.

c-competitiveness. Specifically, an *offline* algorithm for Orkm computes, for each $x_t (t \geq 1)$, the most succinct key relative to the partial context $\{x_1, \ldots, x_t\}$, with the complete knowledge of the entire context $\mathbf{I}$, *i.e.,* also knowing the subsequent instances following x_t in advance. Let $\mathcal{A}$ be any *online* algorithm that is subject to the specification of Orkm. We say that $\mathcal{A}$ is *c-competitive* if for any input of Orkm, *i.e.,* $\mathbf{I}$, M, x_0 and α, for any instance x_t, the succinctness of key E_t computed by $\mathcal{A}$ is no larger than c-times of that by *any offline* algorithm for Orkm. Here c is called the *competitive ratio* of $\mathcal{A}$. Recall that n is the total number of features. We prove the following.

Theorem 4: *There exists no deterministic algorithm for* Orkm *that is $O(n)$-competitive, even the conformity bound α is fixed to 1.* $\square$

Proof: Assume *w.l.o.g.* initially $\mathbf{I}$ and E are empty and all online instances have predictions different from that of x_0. Let the number of online instances be n which is equal to the total number of features. Consider an adversary that controls the arrival order and features of online instances. Let $\mathcal{A}$ be any deterministic online algorithm for ORKM. In each step i, the adversary can always construct the next instance x_{i+1} so that it is identical to x_0 on all those features picked by $\mathcal{A}$ for E_i, while differing on all the other features. Consider offline algorithm OFF that picks the same one feature for which all n instances have distinct values. Then after processing all n instances, $\mathcal{A}$ picks n features for E_n (*i.e.,* succinct$(E_n) = n$) while the succinctness of E_i returned by OFF for all $i \in [1, n]$ is 1. □

Theorem 4 shows that it is impossible to have simple (*i.e.,* deterministic) algorithms that can maintain even relative keys ($\alpha = 1$), while complying with explanation coherence. Nonetheless, we next develop two approaches that mitigate the intractability.

5.2 Randomized Online Key Monitoring

Our first approach is to settle with randomized algorithms. We develop a randomized online algorithm, denoted by OSRK, that coherently updates relative keys with a context that expands with online instances, and is theoretically competitive for relative keys.

A randomized algorithm. Given an instance x_0, any conformity bound $\alpha \in (0, 1]$, a blackbox ML model M, and a stream of instances $x_1, x_2, \ldots$ arriving online that are predicted by M and added to the context $\mathbf{I}$ one by one, algorithm OSRK returns an α-conformant key E_t for x_0 relative to $\mathbf{I}$ upon the arrival of each instance $x_t (t \geq 1)$.

From a birds' eye view, algorithm OSRK expands the relative key E when the incoming instances have predictions different from that of x_0. For each such instance x_t, OSRK grows E with features on which x_t and x_0 do not agree; it does this by adding such features to E independently with a certain probability. During the process, if E does not constitute an α-conformant key relative to the current context $\mathbf{I}$, OSRK continues to double the selection probability of these features until E becomes a valid α-conformant relative key.

We next give the details. As shown in Algorithm 2, upon the arrival of instance x_t, OSRK first updates $\mathbf{I}$ with x_t (line 1). It then checks if its prediction matches that of x_0. If so, it simply returns E_{t-1} as E_t (line 2). Otherwise, it updates the α-conformant key E_t starting from E_{t-1} by expanding it with more features (lines 3-15). Specifically, if x_t is the first online instance such that $M(x_t) \neq M(x_0)$, *i.e.,* $E_{t-1} = \emptyset$, OSRK assigns each feature A_i an initial weight $w_i = 2^{-k}$ where k is the maximum integer for which $2^{-k} < 1/n$, and then adds each feature A_i to E_t with probability w_i (lines 3-6). If there exists no subset $\mathbf{I}^\alpha \subseteq \mathbf{I}$ on which E_t holds as a relative key (recall Section 3.1), OSRK executes the following steps iteratively, until E_t is a valid α-conformant relative key (lines 8-15).

Let set S_t collect all those features on which x_t and x_0 do not agree (lines 7). OSRK first assigns an "aggregated" weight $\mu_t = \sum_{i \in S_t} w_i$ to x_t, *i.e.,* the sum of all the weights w_i corresponding to all the feature collected in S_t (line 9). It then checks if $\mu_k > \log p_t$ where p_t records that x_t is the p_t-th online instance with prediction different from x_0. If so, it picks an arbitrary feature from S_t and adds it to E_t without weight update and confirms that E_t is an α-conformant relative key (lines 10-11). Otherwise, OSRK proceeds with a *weight augmentation* procedure: for each feature A_i collected in S_t, it first doubles its weight w_i if $w_i < 1$; it then adds A_i to E_i with a probability w_i (lines 12-15). It iterates over the entire process again until E_t is confirmed an α-conformant relative key (line 8).

Example 7: Continue with Example 3. Consider a stream of new online loan instances for $\mathbf{I}_0$: x_7 (Female, 3-4K, poor, 2) with prediction "Denied"; x_8: (Male, 3-4K, good, 1) with prediction "Approved"; and x_9: (Male, 3-4K, poor, 0) with prediction "Approved". We know initially the key E for x_0 relative to $\mathbf{I}_0$ is {Income, Credit}. No action is needed for x_7, as its prediction matches that of x_0. x_8 does not change E either as it differs from x_0 on E due to Credit. However, the arrival of x_9 invalidates E as a key, which triggers OSRK to pick Dependent for E as it differs from x_0 in this feature. Hence, it then returns {Income, Credit, Dependent} as the updated E for x_9. □

Correctness & complexity. Algorithm OSRK is correct as (a) by expanding E_t starting from E_{t-1} for each instance x_t, it complies with the explanation coherence condition; and (b) there must exist $\mathbf{I}^\alpha \subseteq \mathbf{I}$ on which E_t holds as a relative key when the condition in line 8 evaluates to *True*, *i.e.,* E_t is α-conformant.

Upon the arrival of each online instance x_t, OSRK returns an α-conformant key relative to the updated context $\mathbf{I}$ in $O(n \log n)$ time, where n is the number of features, *independent of* the number of instances processed, *i.e.,* the size of $\mathbf{I}$. Indeed, (a) each weight augmentation process takes $O(n)$ time; and (b) the number of weight augmentation processes is at most $\log n$. This is because the weight w_i of each feature A_i is initially no larger than $1/n$ and never exceeds 2; once it surpasses 1, A_i is added to E_t for certain. The low complexity of OSRK assures that it does serve efficient explanation monitoring for relative keys at inference time on-the-fly.

Competitiveness. Algorithm OSRK is competitive when maintaining a relative key, with a logarithmic competitive ratio that breaks the lower bound of competitiveness of deterministic algorithms in Theorem 4. Recall that n is the number of features.

Theorem 5: *For each* x_t, OSRK *always returns a* $(\log t \cdot \log n)$-*bounded absolute key for* x_0 *relative to* $\mathbf{I} = \{x_1, \ldots, x_t\}$ *when* $\alpha = 1$. □

Proof: Assume *w.l.o.g.* we have an offline optimal algorithm, denoted by OPT, that always returns the most succinct keys relative to $\{x_1, \ldots, x_t\}$ for each x_t, using full knowledge of the entire $\mathbf{I}$ known in advance. Let k_{OSRK} and k_{OPT}, respectively, be the succinctness of keys computed by OSRK and OPT over time. We observe that there are three stages in OSRK where features can be added to E while serving $x_1, \ldots, x_t$: (1) when handling the first instance that predicts differently from x_0 (line 6); (2) before weight augmentation (line 11); and (3) during weight augmentation (line 15). Denote by random variables X, Y and Z the number of features added to E in these three stages, respectively. k_{OSRK} is naturally the sum of the expectations of X, Y and Z. We prove (1)

$$\mathbb{E}(X) = \sum_{i \in \mathbf{A}} w_i \leq \frac{1}{n} * n = 1.$$

(2) By cumulatively multiplying the probability of each related feature not being added to E previously, we have

$$\mathbb{E}(Y) \leq \sum_{i=1}^{t} \prod_{j \in S_i} (1 - w_j) \leq \sum_{i=1}^{t} (1 - \frac{\mu_i}{|S_i|})^{|S_i|} < \sum_{i=1}^{t} e^{-\ln i} < \ln t + 1;$$

(3) The number of weight augmentation rounds is $O(\log n \cdot k_{\text{OPT}})$ and expected number of added features per round is $O(\log t)$, hence

$$\mathbb{E}(Z) \leq \log t \cdot \log n \cdot k_{\text{OPT}} = O(\log t \log n) k_{\text{OPT}}.$$

Combining $\mathbb{E}(X)$, $\mathbb{E}(Y)$ and $\mathbb{E}(Z)$, we have

$$k_{\mathsf{OSRK}} = \mathbb{E}(X) + \mathbb{E}(Y) + \mathbb{E}(Z) \leq \log t \log n \cdot k_{\mathsf{OPT}}. \qquad \square$$

5.3 A Deterministic Algorithm for Static Features

Our second approach in response to the intractability of Theorem 4 is to consider the class of instances with static features [6, 57], where all x_i's and their predictions by M are known in advance but their order of arrival to **I** is uncertain and is revealed online.

Such instances are brought to our attention by the common practice of interactive product recommendation in e-commercial systems [6, 99], where users and products are typically stable and have static features, *e.g.,* user demographics and product attributes. Hence their pairwise "similarity" scores that are used for generating recommendations can be predicted and materialized offline as a preprocessing. These predictions can then serve real-time on-demand recommendation when a user seeks service, *e.g.,* suggesting products of related categories when a user checks a product.

Mapping to problem Orkm, the extra information we have now is a universe $\mathbb{U}$ of all instances and their predictions and **I** still collects instances from $\mathbb{U}$ that arrive online, one after another as previously defined in Orkm. This seemingly minor extra bit of input enables us to develop a deterministic online algorithm for Orkm with competitiveness comparable to the randomized competitiveness of OSRK (Section 5.2), much better than the lower bound in Theorem 4.

A deterministic algorithm. As a proof, we present such an algorithm, denoted by SSRK (Algorithm 3). Similar to OSRK, it maintains an α-conformant relative key while preserving coherence upon the arrival of each x_t, but it does this deterministically with $\mathbb{U}$.

The core idea is to expand the relative key with potential features, by maintaining (a) a non-increasing potential function Φ based on the yet-to-arrive instances in $\mathbb{U}$ that have predictions different from x_0 and (b) a dynamic importance weight w_i of each feature A_i. Here Φ assists SSRK in deciding the number of new features to be added to the relative key E_t upon the arrival of an instance x_t, and the importance weights help it pick the best ones when E_t grows.

Algorithm SSRK consists of two phases: an offline initialization phase (lines 1-5) and an online phase that updates relative key for x_0 at inference time as instances from $\mathbb{U}$ arrive sequentially (lines 6-17).

Offline initialization. As an offline process, SSRK initializes the *importance weight* w_i for each feature A_i and sets them all to $1/2n$, similar to OSRK (line 1). It also identifies the set U of instances in $\mathbb{U}$ that have predictions different from x_0 (line 2), and assigns an aggregated score μ_j to each instance $\mathsf{x}_j \in U$ based on relevant feature importance weights (lines 3-4). It finally initializes a *potential function* Φ with the aggregated scores of instances in U (line 5).

Online expansion. Upon the arrival of instance x_t, SSRK first adds x_t to **I** and expands E_t starting from E_{t-1} (line 6). If x_t shares the same prediction as x_0, *i.e.,* $\mathsf{x}_t \notin U$, it simply returns E_{t-1} as E_t (line 7). If the current E_t is not a valid α-conformant key, SSRK selects features from S_t (initialized in line 4 for each instance in U), via *weight augmentation* (line 10), and adds them to E_t (lines 8-16).

To do this, it identifies the minimum integer k for which $2^k\mu_t > 1$ (line 9), increases weight w_i by 2^k for each feature A_i in S_t (line 10), and calculates the new potential function value Φ' with the updated feature weights (line 11). Note that $\Phi' > \Phi$. SSRK then iteratively expands E_t guided by Φ and Φ' (lines 12-16). In each iteration, it finds the feature from S_t that minimizes the number of instances from $\mathbb{U}$ that agree with x_0 on the selected features while having a different prediction (line 13). It then updates E_t, S_t, U and Φ' accordingly (lines 14-16). The process terminates when Φ' is no longer greater than Φ (lines 12). Finally, it sets Φ to Φ', to prepare for the arrival of subsequent online instances to **I** (line 17).

ALGORITHM 3: Algorithm SSRK

```
Input: a universe $\mathbb{U}$ of $m$ instances and their predictions by $M$, an
       instance $\mathsf{x}_0$, conformity bound $\alpha \in (0,1]$, and a context $\mathbf{I}$
       that receives a stream of incoming instances $\mathsf{x}_1, \mathsf{x}_2, \ldots$ from
       $U$, arriving one by one.
Output: Upon the arrival of each $\mathsf{x}_t (t \in [1, +\infty))$, an
        $\alpha$-conformant key $E_t$ for $\mathsf{x}_0$ relative to context
        $\mathbf{I} = \{\mathsf{x}_1, \ldots, \mathsf{x}_t\}$ (assume $E_0 = \emptyset$).
Offline initialization:
1  foreach $i \in \mathbf{A}$ do $w_i = 1/2n$ ;   // $\mathbf{A} = \{1, \ldots, n\}$; $i$ refers to feature $A_i$
2  $U \leftarrow \mathbb{U}^c_{M(\mathsf{x}_0)}$ ;   // $\mathbb{U}^c_{M(\mathsf{x}_0)}$ denotes set $\{\mathsf{x} \in \mathbb{U} \mid M(\mathsf{x}) \neq M(\mathsf{x}_0)\}$
3  foreach $\mathsf{x}_j \in U$ do
4      $S_j = \{i \in \mathbf{A} | \mathsf{x}_j[A_i] \neq \mathsf{x}_0[A_i]\}$; $\mu_j = \sum_{i \in S_j} w_i$ ;
5  $\Phi = \sum_{\mathsf{x}_j \in U} m^{2\mu_j}$ ;   // $\Phi$ is a potential function
Upon the arrival of instance $\mathsf{x}_t$:
6  $\mathbf{I} \leftarrow \mathbf{I} \cup \{\mathsf{x}_t\}$; $E_t \leftarrow E_{t-1}$;
7  if $\mathsf{x}_t \notin U$ then return $E_t$;   // no need to expand $E_t$ as $M(\mathsf{x}_t) = M(\mathsf{x}_0)$
8  if $|\bigcap_{j \in E_t} \mathbf{I}[A_j = \mathsf{x}_0[A_j]] \cap \mathbf{I}^c_{M(\mathsf{x}_0)}| > (1-\alpha) \cdot |\mathbf{I}|$ then
9      let $k$ be the minimum integer such that $2^k \mu_t > 1$ ;
10     foreach $i \in S_t$ do $w_i \leftarrow 2^k w_i$;   // weight augmentation
11     update potential function $\Phi' \leftarrow \sum_{\mathsf{x}_j \in U} m^{2\mu_j}$ with new $w_i$'s;
12     while $\Phi' > \Phi$ do
13         $i \leftarrow \arg\min_{i \in S_t} |\bigcap_{j \in E} \mathbb{U}[A_j = \mathsf{x}_0[A_j]] \cap \mathbb{U}[A_i =$
             $\mathsf{x}_0[A_i]] \setminus \mathbb{U}^c_{M(\mathsf{x}_0)}|$;   // $\mathbb{U}[A_i = a]$ is set $\{\mathsf{x} \in \mathbb{U} \mid \mathsf{x}[A_i] = a\}$
14         $E_t \leftarrow E_t \cup \{i\}$; $S_t \leftarrow S_t \setminus \{i\}$;
15         $U \leftarrow \bigcap_{j \in E_t} \mathbb{U}[A_j = \mathsf{x}_0[A_j]] \cap \mathbb{U}^c_{M(\mathsf{x}_0)}$;
16         update potential $\Phi' \leftarrow \sum_{\mathsf{x}_j \in U} m^{2\mu_j}$ with the new $U$;
17     $\Phi \leftarrow \Phi'$;
```

Complexity. As a one-off cost, the offline initialization phase of SSRK takes $O(nm)$-time. For each of the m online instances, SSRK takes $O(nm)$ time to update the α-conformant relative key.

Competitiveness. Despite Theorem 4, SSRK is competitive for instances with static features, comparable to the randomized OSRK.

Theorem 6: *For each* x_t*,* SSRK *always returns a* $(\log m \cdot \log n)$*-bounded key for* x_0 *relative to* $\mathbf{I} = \{x_1, \ldots, \mathsf{x}_t\}$ *when* $\alpha = 1$. $\square$

Proof sketch: Denote by OPT the offline optimal algorithm that always returns the most succinct keys for each x_t, with the entire sequence of **I** known in advance. Let k_{SSRK} and k_{OPT}, respectively, are the succinctness of keys computed by SSRK and OPT over time. Observe that features can only be added to relative keys by SSRK after the weight augmentation (line 10). To compute k_{SSRK}, we only need to determine the number of times weight augmentation occurs and the number of features added to the key after each weight augmentation (lines 11-16), denoted by X and Y, respectively.

(1) We first show that $X \leq \log 2n \cdot k_{\mathsf{OPT}}$. To do this, we prove that (a)

$\mu_t \leq 1$; (b) SSRK undergoes at most $\log(2n)$ weight augmentations when $\mu_t \leq 1$; and (c) at least one feature in OPT is involved when performing weight augmentation for x_t.

(2) We then prove that $Y \leq \log m$. The idea of the proof for this is that we need to ensure that Φ does not increase when selecting features. We know only instances in U can contribute to Φ. Consider *w.l.o.g.* an instance $\mathrm{x}_p \in U$. Before the weight augmentation, the contribution of x_p to Φ is $m^{2\mu_p}$. Let Δ_p be the increment in the aggregated score of x_p after the weight augmentation. This means the contribution of x_p to Φ is $m^{2(\mu_p+\Delta_p)}$ if x_p is still in U. We prove that after at most $\log m$ feature selections, x_p is no longer in U. This indicates that the contribution of x_p to Φ will be decreased compared to before the weight augmentation. This also applies to all other instances in U, ensuring a non-increasing Φ. Hence, $Y = O(\log m)$.

Combing X and Y, we have $k_{\mathsf{SSRK}} = X \cdot Y \leq \log m \log n \cdot k_{\mathsf{OPT}}$. □

As will be shown shortly in Section 7.4, in practice SSRK often outperforms OSRK in the quality of relative keys that they monitor.

6 CLIENT-CENTRIC FEATURE EXPLANATION

Built upon the algorithms in Sections 4-5, we develop CCE, a proof-of-concept that offers client-centric feature explanation.

Overview. CCE sits between a possibly remote ML model M and a client user that uses predictions from M. It returns relative keys as feature explanations by collecting inference instances and their predictions at the client-side as the context. During the process, CCE is completely controlled by the client-side user and does not access the model M as existing approaches do. Therefore, it gives the right to explain to model users and works for ML models that do not allow explanation queries, *e.g.,* cloud ML services that restrict or charge model accesses and do not support feature explanation [9].

Targeted users. CCE targets two types of model users: (a) users who do batch inference, *e.g.,* ML researchers or applications doing model inference over an inference set; and (b) users with long-term high demand for third-party ML inference services, *e.g.,* banks that use ML-based credit or loan assessment models [11, 12] (Example 1). CCE either uses their inference set or gathers their inference instances during model serving as the context of relative keys.

Handling context. Depending on whether the client has the capacity to hold all inference instances as the context **I** in its entirety, users may opt to use CCE in the *batch mode*, under which it computes relative keys as feature explanations via algorithm SRK with access to the complete **I** as input. When the client has limited capacity or **I** is not readily available in time, CCE offers to operate in the *online mode*, under which it treats **I** as a stream of inference instances arriving online and invokes algorithm OSRK to compute and update relative keys upon the arrival of each inference instance in **I**.

Remark. (1) CCE is not designed for one-time ad-hoc ML users, *e.g.,* a particular loan applicant (customer) of a bank who has no context.

(2) One can also rank the features in the relative keys by the order they are picked by algorithm SRK (OSRK in the online mode). However, we remark that relative keys are not a replacement of feature importance explanations that demand much more (see Section 8).

7 EXPERIMENTAL STUDY

We experimentally evaluate the benefits of relative keys, and the effectiveness and efficiency of CCE. We start with the settings.

7.1 Experimental Setup

Datasets. We used 9 real-world datasets as summarized in Table 1. Among them, 5 are from the UCI ML repository [7] and Kaggle contest portal [3] for evaluating general ML classification tasks and their explanations [48, 75, 76, 80]. The other 4 are specifically built for the evaluation of ML-based entity matching and its explanations [32, 70]. Different from datasets for general ML tasks, each dataset for entity-matching is a pair of tables and the ML task is to link entity instances across the two tables.

Compared methods. We compared with 7 state-of-the-art explanation methods, with publicly available implementations (Table 2).

(a) Feature explanation. Anchor and Xreason are popular feature explanation methods. In particular, Xreason always computes formal explanations; however, it requires to know the structure of the ML models and works for (an ensemble of) decision trees only. Anchor is a heuristic feature explanation method and is not limited to decision trees. To use it, one needs to specify a threshold parameter that heuristically guides the chances of sampled instances conforming to explanations. By adjusting the threshold via a trial-and-error process, we can roughly control the size of Anchor's explanations.

(b) Feature importance. LIME, SHAP and GAM are feature importance based methods, *i.e.,* they assign each feature an importance score. Following [13], we derived feature explanations from importance scores by first ranking features in descending order according to their importance scores and then picking features one by one until a feature explanation of targeted size threshold is formed.

(c) Pattern-level explanation. We used IDS, a method that generates pattern-level explanations in the form of independent conjunctive rules over features that together summarize a given dataset. As remarked in Section 2, it is a global explanation method.

(d) Entity-matching explainer. We also compared with CERTA, a feature importance based method specialized for entity-matching by making use of the specific semantics of entity-matching tasks. It has been shown superior in explanation quality than general feature importance explanation methods [95], including LIME and SHAP.

All methods, except IDS, compute local explanations, *i.e.,* they explain a given instance, while IDS generates pattern-level explanations that summarize the entire dataset. All compared local methods, except Xreason, have a variable size of feature explanations determined by their threshold parameter. Moreover, they all need to query ML models M when deducing explanations, by invoking M over generated neighbor instances in the feature space. In contrast, CCE has no access to M. In our tests, all compared methods are tested with their default configurations unless stated otherwise.

Measures. We tested the efficiency of all methods by their average running time for explaining a single instance. We used five measures to quantify the quality of their feature explanations.

(a) Conformity [33]. The conformity of a method is the percentage of its explained instances whose feature explanations are conformant.

(b) Precision. The precision of a feature explanation E for an instance

General ML datasets and their prediction tasks			
datasets	**#instance**	**#feature**	**task**
Adult [53]	32,526	14	predict whether an adult earns ≥ 50K a year
German [36]	1,000	21	German credit data; classify credit levels
Compas [2]	6,172	11	score criminal defendant's likelihood of re-offending based on the COMPAS algorithm
Loan [4]	614	11	predict whether to approve a loan application
Recid [87]	6,340	15	predict recidivism for individuals released from North Carolina prisons in 1978 and 1980

Entity-matching datasets (each is a pair of datasets)				
datasets	**#instance**	**#match**	**#feature**	**domain**
A−G (Amazon-Google)	11,460	1,167	3	Software
D−A (DBLP-ACM)	12,363	2,220	4	Citations
D−G (DBLP-GoogleScholar)	28,707	5,347	4	Citations
W−A (Walmart-Amazon)	10,242	962	5	Electronics

Table 1: Summary of 9 datasets and their statistics

General local explanation methods	
methods	**description**
LIME [75]	a linear classifier trained on perturbed neighbor instances
SHAP [61]	assign a game-theory based importance score for each feature
Anchor [76]	if neighbors share features, their predictions will *likely* be the same
GAM [60]	use Generalized Additive Models (GAM) to determine how feature value changes affect ML predictions
Xreason [48]	compute formal explanation over entire feature space via Maximum Satisfiability (MaxSAT); works for white-box decision trees only
Pattern-level explanation method	
method	**description**
IDS [56]	generate a set of independent rules that summarizes a given dataset
Entity-matching explanation method	
method	**description**
CERTA [95]	use a probabilistic framework to produce explanations for Entity Matching by evaluating the outcomes of perturbed records

Table 2: State-of-the-art explanation methods

x is the maximum ratio α such that E is an α-conformant key relative to the set of all instances explained. The precision of a method is the average precision of explanations it generated.

(c) Recall. We use recall to measure the "coverage" of methods that produce conformant explanations, *i.e.,* Xreason and CCE. For an instance x, let E and E' be the conformant explanations of Xreason and CCE, respectively. Let $D(E)$ be the set of instances that agree with and conform to E; similarly for $D(E')$. Then the recall of Xreason (resp. CCE) is $|D(E)|/|D(E) \cup D(E')|$ (resp. $|D(E')/|D(E) \cup D(E')|$).

(d) Succinctness: the number of features in an explanation.

(e) Faithfulness [20]. It assesses how well an explanation reflects a model's predictive capability. Consider an instance x and its perturbed version x′, generated by masking [20] features deemed impactful to model outcomes by explanation method $\mathcal{A}$; for feature explanation methods, these are the features included in their explanations. The faithfulness of $\mathcal{A}$ *w.r.t.* a set D of instances explained is $\sum_{x \in D} I(M(x) = M(x'))/|D|$, where I is the indicator function.

A lower faithfulness value indicates that perturbing features identified by $\mathcal{A}$ impacts model predictions more significantly. That is, the explanation generated by $\mathcal{A}$ is indeed more faithful to the model's inner workings, *i.e.,* it has higher quality.

Note that the size of derived feature explanations by feature importance methods (SHAP, LIME, GAM and CERTA) and that of Anchor depend on their threshold parameters. Hence, when evaluating conformity, precision and faithfulness we fix the size of their feature explanations to the same as that of CCE, using their corresponding parameters. When evaluating recall and succinctness, we only compare with formal method (Xreason) as all other methods have no conformity guarantees and can be arbitrarily small in succinctness and high in recall by simply returning only one feature or even empty (which would have zero conformity and precision).

For conformity, precision and recall, the closer to 100% the better; in contrast, for succinctness and faithfulness, the lower the better.

Configuration. We deployed CCE in a machine with an i7-6700HQ CPU @2.6GHz and 32 GB of RAM. Each entity-matching dataset has its own built-in training and inference sets; we used the training set to train Ditto [58], a transformer-based DNN model that uses fine-tuned pretrained embeddings and optimizations specific for entity matching. For every other dataset, a 70/30 ratio was used to split it into training set and inference set. For each training set, we trained the ML model using XGBoost [30], an advanced learner that ensembles multiple base decision-tree classifiers [29, 33]; this is the most complex model still supported by Xreason. For each dataset, we randomly picked 100 instances from their inference set to explain. By default, CCE uses SRK with $\alpha = 1$ and the inference set as its context. Each test was run three times; the average is reported.

7.2 Case Study and User Study

Continuing with Examples 1-2, we first demonstrate the benefits of relative keys as a feature explanation via a case study over the Loan dataset. We also conducted a user study on how users perceive the quality of explanations in different forms from different methods.

Case study. As shown in Examples 1 and 2, the relative key computed by CCE for x_0 in the Loan dataset (a) avoids the heuristic issue of Anchor (*e.g.,* x_1 of Fig. 1) by upholding perfect conformity as Xreason does over inference instances; and (b) is more succinct than the explanation of Xreason, and hence has better recall, *e.g.,* covering x_2 that Xreason is not able to. Moreover, (c) CCE is orders of magnitude faster than Xreason to explain x_0. This illustrates the benefit of CCE and relative key over formal Xreason and heuristic Anchor.

Below we complete the case study by comparing with (a) feature importance explanations by LIME, SHAP and GAM in their native form and (b) pattern-level explanations computed by IDS.

Feature importance explanations. Table 3 shows the feature importance explanations of LIME, SHAP and GAM for the same Loan instance x_0 as in Example 1. The explanations assign each feature of x_0 an importance score. A positive (negative) score means that the feature has a positive (negative) impact on the model prediction of x_0; a larger absolute value of the score implies a greater impact. Note that we do not have an explanation from CERTA as it is specialized for entity-matching tasks and cannot explain x_0 of Loan.

We compare them with feature explanations by Anchor and CCE. By following [13], we convert the feature importance explanations to feature explanations of size 2 (since both the explanations of Anchor and CCE are of size 2): we pick 2 features with the highest impact on the prediction of x_0 according to the scores.

As highlighted in bold in Table 3, the derived feature explanations of LIME and SHAP are exactly the same as that of Anchor, hence sharing the same issue of Anchor due to lack of conformity, *e.g.,* x_1 of Fig. 1 in Loan has the same `Dependents` and `Credit` as x_0 but with a prediction different from that of x_0. The case for GAM is similar.

Pattern-level explanations. We compared with pattern-level explanations. Since pattern-level explanations, including IDS, are global explanations (recall Section 2), they cannot target a given instance,

x_0:	Gender Male	Married No	Dependents 1	Employed No	Edu NotGrad.	Income 3-4K	CoIncome 2-3K	Credit poor	LoanAmount 12-14K	LoanTerm 180	Area Urban
LIME	0.01	-0.02	**-0.11**	-0.01	0.01	-0.01	-0.01	**-0.22**	0.01	0.01	0.01
SHAP	-0.09	-0.35	**-0.41**	-0.05	0.07	-0.04	0.16	**-0.57**	0.04	0.2	-0.4
GAM	0.01	-0.03	-0.01	-0.01	-0.07	0.02	0.14	**-0.39**	-0.01	-0.01	**-0.2**

Table 3: Feature importance explanations for x_0 in Loan

e.g., x_0. To this end, we first specify the `size` parameter of IDS so that it summarizes Loan via 8 rules, as shown below:

IF `LoanAmount`='8-10K'∧`Credit`='good' **Then** Prediction ='Approved'
IF `CoIncome`='1-2K'∧`Credit`='good' **Then** Prediction ='Approved'
IF `LoanTerm`=360∧`Credit`='poor' **Then** Prediction ='Denied'
IF `Income`='1-2K'∧`Credit`='good' **Then** Prediction ='Approved'
IF `Gender`='Female'∧`Credit`='good' **Then** Prediction ='Approved'
IF `Income`='>10K'∧`CoIncome`='0-1K' **Then** Prediction ='Approved'
IF `CoIncome`='0-1K'∧`Credit`='poor' **Then** Prediction ='Denied'
IF `Income`='4-5K'∧`Credit`='good' **Then** Prediction ='Approved'

One can see that the patterns do not explain x_0. Indeed, none of conditions of the 8 rules holds on x_0; therefore, they provide no insight on why x_0 has prediction "Denied". To rectify this, we further run IDS without the restriction on `size`, which yields a set of 1399 rules for Loan and this took IDS 120,000 ms. We verified that among these rules, x_0 is covered by a rule exactly the same as the relative key for x_0 written in rule (Fig. 1). This shows that pattern-level explanations are not flexible to explain a given instance as local explanations and relative keys do, and can be too slow.

User study. We further conducted a user study to assess (a) which form of explanations is easier for users to understand, feature explanation or feature importance, and which method is best perceived; and (b) whether conformity is critical for maintaining user's trust.

To this end, we designed three surveys, accessible online at S_a[1], S_b[2], and S_c[3]. S_b consists of 4 pages, while both S_a and S_c have 2 pages each. The first page of all surveys asked about the participants' background *e.g.,* familiarity with ML. We recruited 24 users familiar with ML terminologies. Each survey had an equal number of participants, and each participant was allowed to participate in the study only once. We used the Loan dataset and presented explanations in the form as used in the case study. We ensured that the participants were unaware of which method generated the explanations.

Design. The study aimed to gather user preferences over four tasks.

(a) SHAP **vs.** CCE. This aims to assess if users find feature explanations (*e.g.,* CCE) easier to understand than feature importance explanations (*e.g.,* SHAP). Users first browsed 5 instances with their corresponding explanations: S_a[P2] (page 2 of S_a) displayed SHAP explanations, while S_b[P2] showcased CCE's explanations. Users were then asked to predict the model's behavior on 10 new randomly sampled instances to see if they got the model's logic from the explanations. We also tracked the time users took to predict.

(b) IDS **vs.** CCE. We also assess whether users prefer local explanations over global pattern-level explanations. For a given instance shown in S_b[P3] , we displayed results from CCE, partial IDS, and full IDS side by side, and asked users which one better helped their

[1]https://forms.gle/HBQLw1yqnpvuAgga8
[2]https://forms.gle/Xm3LC9nveGyBS7TEA
[3]https://forms.gle/Piud7j2vifkrCWgZ8

methods	Adult	German	Compas	Loan	Recid
CCE (SRK)	**11**	**7**	**9**	**7**	**8**
LIME	264	207	167	97	345
SHAP	344	276	154	101	360
Anchor	455	469	201	110	547
GAM	259	36	39	27	204
Xreason	2236	3480	905	443	2154

Table 4: Average time (ms) for computing explanations

understanding of the reasons for model decisions.

(c) Anchor **vs.** CCE. It decides if users are concerned about explanations that are not conformant. In S_b[P4], we presented users with instances where Anchor violated conformity, like x_1 in Fig. 1. We also highlighted that such inconsistencies would not occur with CCE. Using a 5-point Likert scale, we inquired if users felt that Anchor and CCE would bolster their trust in the system.

(d) Xreason **vs.** CCE. When perfect conformity is warranted, we test if more succinct feature explanations are easier for people to comprehend, by using S_b[P2] and S_c[P2] via the same strategy in (a).

Results. From the feedback gathered, we found the following.

(1) People were quicker and more accurate to make predictions for unseen instances using rule-based feature explanations (CCE) than with feature importance explanations (SHAP). Specifically, 72.5% of predictions were correct with feature explanations, while only 52.5% with feature importance. Additionally, the time taken for the latter was twice as the former. This indicates that users find it much easier to understand model behaviors based on feature explanations.

(2) A significant 87.5% of users believed that local rule-based feature explanations are easier to understand than global ones, possibly due to the visual overhead of global rule explanations.

(3) When exposed to explanations that are not conformant, only 25% of users trusted the system (*i.e.,* scoring ≥4 on a 5-point Likert scale). But with perfect conformity, trust rose to 87.5%. This highlights the importance of conformant explanations for user trust.

(4) When provided with conformant explanations, CCE users took 42.9% less time and were 17.5% more accurate than those using Xreason to make decisions. This is likely because Xreason includes too many features, making it challenging for users to quickly parse the conjunctions. It confirms our hypothesis: more succinct explanations do help users understand model decision boundaries better.

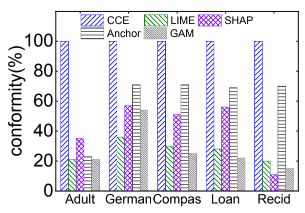
7.3 Feature Explanation: Efficiency and Quality

We next systematically evaluated the efficiency and quality of all methods. We focus on general local explanation methods since IDS does not explain given instances as shown earlier and CERTA works only for entity matching explanation (will cover in Section 7.5).

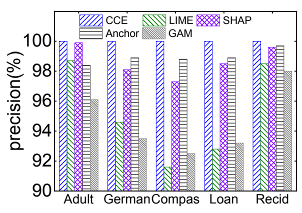
Efficiency. We tested the time that each method takes to explain an instance. The results of all methods are shown in Table 4. On average over all datasets, CCE is 25.8, 29.4, 43.0, 12.5, 226.4 times faster than LIME, SHAP, Anchor, GAM and Xreason, respectively.

Quality. We next evaluated the quality of the feature explanations using all five measures. For feature importance methods, we derived feature explanations from their importance scores as per Section 7.1.

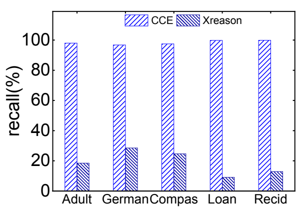
(1) Conformity and precision. We tested the conformity and precision of all methods. The results over all datasets are depicted in Figures 3a-3b. Both the conformity and precision of CCE are consistently



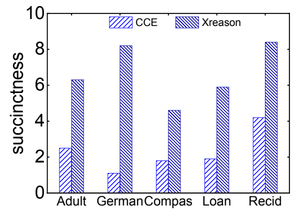
(a) conformity



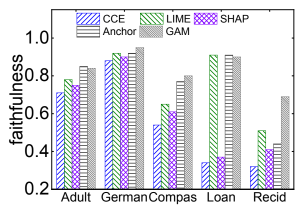
(b) precision



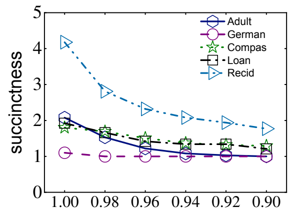
(c) recall



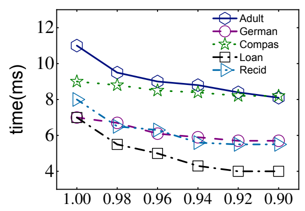
(d) succinctness (*lower the better*)



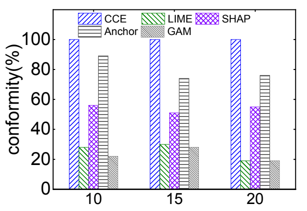
(e) faithfulness (*lower the better*)



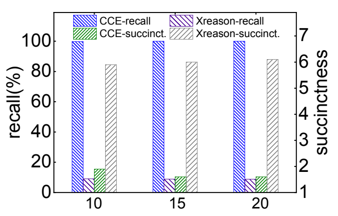
(f) succinctness with varying α



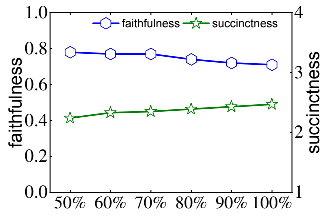
(g) time with varying α



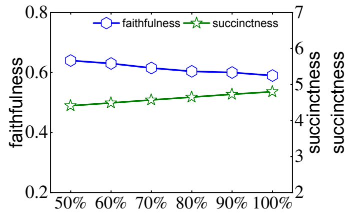
(h) Varying #-bucket



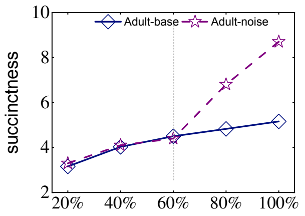
(i) Varying #-bucket



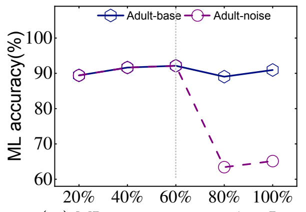
(j) SRK: varying context size



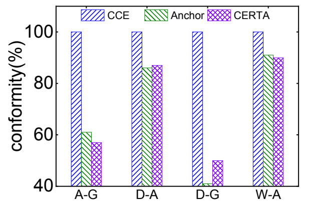
(k) OSRK: varying context size



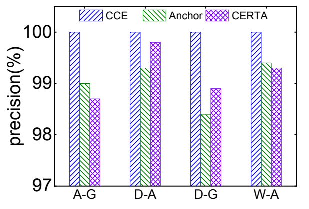
(l) succinctness: varying **I**%



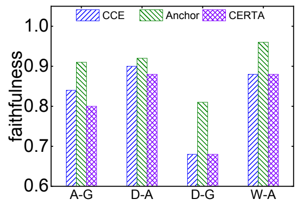
(m) ML accuracy: varying **I**%



(n) Entity-matching explanation



(o) Entity-matching explanation



(p) Entity-matching explanation

Figure 3: The quality and benefits of relative keys

100% in all cases, confirming that relative keys strictly enforce the semantics of explanations for all inference instances. The same also applies to formal explanations by Xreason (not shown). In contrast, LIME, SHAP, Anchor and GAM are all heuristic in nature, and hence cannot warrant conformant explanations.

(2) Recall and succinctness. We next tested the succinctness and recall of methods that produce conformant explanations, *i.e.,* Xreason and CCE. As discussed in Section 7.1, all other methods cannot control the conformity of explanations and hence cannot be compared.

While both returning conformant explanations over the inference set, the explanations of Xreason are much less succinct and significantly lower than CCE in recall. As shown in Fig. 3c, the recall of SRK is consistently above 96.8% in all cases, while it is only 18.5%, 28.5%, 24.6%, 9.1% and 12.9%, respectively, for Xreason over Adult, German, Compas, Loan and Recid, respectively. This is because formal explanations by Xreason are much larger than relative keys of CCE, *e.g.,* on average 2.9 times larger, making them closer to as if returning all features of an instance as the explanation (Fig. 3d).

(3) Faithfulness. We also tested the faithfulness of the explanation methods. As shown in Fig 3e, CCE has the best (*i.e.,* lowest) faithfulness values among all methods over every dataset, consistent with other quality measures. This is somewhat a bit surprising as faithfulness concerns model behaviors on perturbed instances from the feature space but not necessarily in the dataset, even more so for the inference set which CCE uses as the context for relative keys. The positive result of CCE in faithfulness suggests that relative keys to some extent can already capture model behaviors beyond the scope of the context and even the training set.

We did not report the result of Xreason. This is because faithfulness measures the impact of perturbations in features identified by the explanations on the model prediction (Recall Section 7.1). Hence, it is critical to ensure all methods alter the same number of features. As a result, we cannot include Xreason in the tests as its explanations are much larger than others and are not tunable in size.

Flexible trade-offs. We examined the benefit of α-conformant relative keys. By varying the conformity bound α from 1 to 0.9, we evaluated its impact on the performance of CCE. As shown in Figures 3f-3g, we can observe (a) consistent with its theoretical guarantees, SRK is able to rigorously trade conformity for more succinct explanations (Fig. 3f). Over all datasets, on average the succinctness of the relative keys improves (decreases) from 2.2 to 1.3 when α decreases from 1 to 0.9. (b) Relaxing the conformity bound also enables faster computation of the relative keys. As shown in Fig. 3g, CCE is 1.8 times faster when α decreases from 1 to 0.9 over Loan. These show that SRK allows flexible trade-offs between the precision of explanations and performance, *i.e.,* efficiency, succinctness.

Impact of numerical features. We also evaluated the impact of bucketing for numerical features on the quality of explanations. Denote by #-bucket the number of buckets we partition a feature. Varying #-bucket for the `LoanAmount` feature of the Loan dataset from 10 to 20, we tested the quality of feature explanations of all methods. The results for conformity, recall and succinctness are shown in Figures 3h- 3i. The conformity of relative keys is stable against varying #-bucket, while Anchor and feature importance based methods are more sensitive to the changes. The recall and succinctness of CCE and Xreason are both stable *w.r.t.* #-bucket, partially due to their strong enforcement on conformity of the explanations.

Impact of context. We examined the impact of the size $|\mathbf{I}|$ of context $\mathbf{I}$ on the quality of relative keys. Varying $|\mathbf{I}|$ from 50% to 100% of the inference set of Adult, we tested the faithfulness and succinctness of CCE. As shown in Fig. 3j, with larger context, faithfulness of CCE becomes better (decreases), as expected; similarly, relative keys become more succinct with larger context. This said, even with 50% of the inference set, CCE already achieves nearly 90% of the quality of relative keys computed with full inference set. It shows that CCE can operate decently even with limited context.

7.4 Online Explanation Monitoring

We evaluated the effectiveness of CCE for explanation monitoring.

Quality and efficiency. We used all instances in the inference set of each dataset and fed them to the context $\mathbf{I}$ of algorithms OSRK and SSRK of CCE, one instance per step. We used the entire inference set as the universe $\mathbb{U}$ for SSRK. On average over all 5 datasets, it takes OSRK 0.02ms to update the relative key per each online instance added to $\mathbf{I}$, while SSRK takes 0.03ms. SSRK produces more succinct relative keys, *e.g.,* 4.0 vs. 4.9 of OSRK on average.

Impact of context. We also tested the impact of the size of the context. The results are shown in Fig. 3k and are similar to Section 7.3.

An application. As an application of relative key monitoring, we show that CCE (OSRK) can be used to monitor the performance (accuracy) of blackbox models M during model serving.

Specifically, for each dataset, we generated two versions: *base* and *noise*. The base version uses the setup above, *i.e.,* instances are fed to $\mathbf{I}$ one by one. In the noise version, we added randomly generated noise instances to the last 40% of instances, to trigger a dip of the accuracy of M. Denote by $\mathbf{I}\%$ the percentage of inference instances arrived and processed by CCE. Then the succinctness of CCE over base and noise Adult with varying $\mathbf{I}\%$ is shown in Fig. 3l, which shows that the succinctness of the monitored relative keys "abnormally" increases from $\mathbf{I}\% = 60\%$, at which point noise is added to the online instances. This is consistent with the actual accuracy of M as shown in Fig. 3m, which does significantly drop when $\mathbf{I}\% = 60\%$.

7.5 Application: Entity Matching Explanation

As an application, we finally evaluated the effectiveness of CCE for explaining entity matching results. Entity matching is a classic operation that identifies and links records from different data sources referring to the same entity [24]. Different from typical ML tasks, it involves at least two sources and deals with entities with intricate semantic descriptions. Specialized ML-based approaches have been proposed [24, 70] and explaining them becomes imperative [70, 95].

Setup. We used the entity-matching datasets in Table 1. We evaluated the efficiency and quality of CCE, Anchor and CERTA, a specialized state-of-the-art entity matching explainer (Table 2). Xreason does not work as the entity matching model is a DNN (recall configuration in Section 7.1). We did not report results of general feature importance methods since it has already been shown that CERTA is superior than *e.g.,* SHAP and LIME in quality for the task [95].

Quality. As shown in Figures 3n-3o, CCE consistently achieves 100% for both conformity and precision in all cases. In contrast, on average over the four datasets, conformity of CERTA and Anchor is 71.0% and 69.8%, respectively, while precision is 99.2% and 99.0%. This shows that CCE warrants conformity, independent of ML task.

As depicted in Fig 3p, CCE outperforms Anchor in faithfulness over all datasets, *e.g.,* better than Anchor by 16.1% on D−G. While CCE's faithfulness is slightly behind CERTA on the A−G and D−A datasets, it is on par with CERTA on D−G and W−A. This is already encouraging as CERTA is a specialized entity-matching explainer and exploits task specific optimizations [95] not available to CCE.

Efficiency. CCE, as a data-driven approach, is 4 orders of magnitude faster than CERTA on average across four datasets. This verifies the versatility of CCE for *e.g.,* entity matching tasks.

7.6 Summary

We find the following on average. (1) CCE is 60.7%, 3.1% and 24.6% better than heuristic methods in conformity, precision and faithfulness, respectively, and is 79.7% and 2.9x better than formal method in recall and succinctness. (2) CCE is faster than every method on every dataset, by 2 orders of magnitude, up to 497.1 times. (3) CCE has comparable quality to specialized entity-matching explainer over nearly half of test cases, 10.4% higher than heuristic methods (formal method does not even work); moreover, it is 4 orders of magnitude faster. (4) CCE supports flexible trade-off between explanation quality and efficiency, and model performance monitoring. (5) Case study and user study verify the benefits of relative keys.

8 CONCLUSION

Heuristic feature explanation methods are fast but can be problematic due to lack of conformity guarantees. On the other hand, formal methods strictly maintain conformity of their explanations but are intractable to compute and work for decision trees only. We propose relative keys that have the best of both worlds. Relative keys associate explanations with a context and enforce strict conformity of explanations over the context. Based on it, we develop CCE, a client-centric feature explanation framework that assumes no model access. Using real-life datasets, we show that CCE is 2 orders of magnitude faster than existing methods and superior in the quality of its feature explanations under various quality measures.

One topic for future work is to extend relative keys for feature importance based explanations, by extending the notion and computation of Shapley values to the online setting with a dynamic context. Another topic is to revisit global pattern-level explanations relative to a context, which would require novel techniques to approximate global patterns with dynamically observed context.

REFERENCES

[1] Machine learning best practices in financial services. https://d1.awsstatic.com/whitepapers/machine-learning-in-financial-services-on-aws.pdf, 2020.

[2] Compas dataset. https://www.kaggle.com/datasets/danofer/compass, 2022.

[3] Kaggle. https://www.kaggle.com/, 2022.

[4] Loan dataset. https://www.kaggle.com/datasets/vikasukani/loan-eligible-dataset, 2022.

[5] Manual review. https://seon.io/resources/dictionary/manual-reviews/, 2022.

[6] online predictions with static features. https://www.zdnet.com/article/machine-learning-is-going-real-time-heres-why-and-how/, 2022.

[7] Uci machine learning repository. https://archive.ics.uci.edu/ml/index.php, 2022.

[8] Unleashing the power of machine learning models in banking through explainable artificial intelligence (xai). https://www2.deloitte.com/us/en/insights/industry/financial-services/explainable-ai-in-banking.html, 2022.

[9] Amazon sagemaker. https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-model-explainability.html, 2023.

[10] Explainable artificial intelligence: From static to dynamic. https://sites.google.com/view/dynxai-ecmlpkdd-2023, 2023.

[11] Upstart. https://www.upstart.com/, 2023.

[12] Zest ai. https://www.zest.ai/, 2023.

[13] D. Afchar, V. Guigue, and R. Hennequin. Towards rigorous interpretations: a formalisation of feature attribution. In *ICML*, 2021.

[14] C. Agarwal, S. Krishna, E. Saxena, M. Pawelczyk, N. Johnson, I. Puri, M. Zitnik, and H. Lakkaraju. Openxai: Towards a transparent evaluation of model explanations. In *NeurIPS*, 2022.

[15] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[16] E. Albini, A. Rago, P. Baroni, and F. Toni. Relation-based counterfactual explanations for bayesian network classifiers. In *IJCAI*, pages 451–457, 2020.

[17] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 100–105. ACM, 2003.

[18] L. Amgoud and J. Ben-Naim. Axiomatic foundations of explainability. In *IJCAI*, pages 636–642, 2022.

[19] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[20] P. Atanasova, J. G. Simonsen, C. Lioma, and I. Augenstein. A diagnostic study of explainability techniques for text classification. In *EMNLP*, pages 3256–3274, 2020.

[21] G. Audemard, F. Koriche, and P. Marquis. On tractable XAI queries based on compiled representations. In *KR*, pages 838–849, 2020.

[22] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, 1999.

[23] A. Baraldi, F. Del Buono, M. Paganelli, F. Guerra, et al. Using landmarks for explaining entity matching models. In *EDBT*, volume 2021, pages 451–456, 2021.

[24] N. Barlaug and J. A. Gulla. Neural networks for entity matching: A survey. *TKDD*, 15(3):1–37, 2021.

[25] G. Blanc, J. Lange, and L. Tan. Provably efficient, succinct, and precise explanations. In *NeurIPS*, pages 6129–6141, 2021.

[26] R. Bommasani, D. A. Hudson, E. Adeli, R. B. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. S. Chatterji, A. S. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[27] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, 2001.

[28] P. Buneman and W. C. Tan. Provenance in databases. In *SIGMOD*. ACM, 2007.

[29] N. Burkart and M. F. Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[30] T. Chen, T. He, M. Benesty, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[31] M. C. Cooper and J. Marques-Silva. On the tractability of explaining decisions of classifiers. In *CP*, volume 210, pages 21:1–21:18, 2021.

[32] S. Das, A. Doan, C. G. Psgc, and P. Konda. The magellan data repository. 2016.

[33] S. Dasgupta, N. Frost, and M. Moshkovitz. Framework for evaluating faithfulness of local explanations. In *ICML*, pages 4794–4815. PMLR, 2022.

[34] D. Deutch and N. Frost. Constraints-based explanations of classifications. In *ICDE*, pages 530–541, 2019.

[35] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[36] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *KDD*, pages 259–268. ACM, 2015.

[37] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[38] A. Garivier and E. Moulines. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, pages 174–188, 2011.

[39] K. E. Gebaly, P. Agrawal, L. Golab, F. Korn, and D. Srivastava. Interpretable and informative explanations of outcomes. *Proc. VLDB Endow.*, 8(1):61–72, 2014.

[40] Z. Geng, M. Schleich, and D. Suciu. Computing rule-based explanations by leveraging counterfactuals. *Proc. VLDB Endow.*, 16(3):420–432, 2022.

[41] N. Gorji and S. Rubin. Sufficient reasons for classifier decisions in the presence of domain constraints. In *AAAI*, pages 5660–5667, 2022.

[42] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40. ACM, 2007.

[43] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.

[44] D. Gunning and D. Aha. Darpa's explainable artificial intelligence (xai) program. *AI magazine*, 40(2):44–58, 2019.

[45] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G.-Z. Yang. Xai—explainable artificial intelligence. *Science robotics*, 4(37):eaay7120, 2019.

[46] J. Haug, A. Braun, S. Zürn, and G. Kasneci. Change detection for local explainability in evolving data streams. In *CIKM*, pages 706–716, 2022.

[47] A. Ignatiev. Towards trustable explainable AI. In *IJCAI*, pages 5154–5158, 2020.

[48] A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *AAAI*, pages 3776–3785, 2022.

[49] A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.

[50] A. Ignatiev, N. Narodytska, and J. Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019.

[51] D. Janzing, L. Minorics, and P. Blöbaum. Feature relevance quantification in explainable AI: A causal problem. In *AISTATS*, 2020.

[52] A. Karimi, G. Barthe, B. Balle, and I. Valera. Model-agnostic counterfactual explanations for consequential decisions. In *AISTATS*, 2020.

[53] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 202–207. AAAI Press, 1996.

[54] S. Korman. On the use of randomization in the online set cover problem. *Weizmann Institute of Science*, 2, 2004.

[55] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137, 2015.

[56] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *SIGKDD*, pages 1675–1684, 2016.

[57] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu. Multi-task representation learning for travel time estimation. In *KDD*, pages 1695–1704. ACM, 2018.

[58] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60, 2020.

[59] Q. V. Liao, Y. Zhang, R. Luss, F. Doshi-Velez, and A. Dhurandhar. Connecting algorithmic research and usage contexts: A perspective of contextualized evaluation for explainable ai. In *AAAI*, volume 10, pages 147–159, 2022.

[60] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *KDD*, pages 150–158. ACM, 2012.

[61] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017.

[62] J. Marques-Silva and A. Ignatiev. Delivering trustworthy AI through formal XAI. In *AAAI*, pages 12342–12350, 2022.

[63] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *Proc. VLDB Endow.*, 4(1):34–45, 2010.

[64] M. Miao. Debating the right to explanation: An autonomy-based analytical framework. *SAcLJ*, 34:864, 2022.

[65] Z. Miao, Q. Zeng, B. Glavic, and S. Roy. Going beyond provenance: Explaining query answers with pattern-based counterbalances. In *SIGMOD*, pages 485–502. ACM, 2019.

[66] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[67] C. Molnar. *Interpretable machine learning*. 2020.

[68] R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.

[69] C. Mougan, K. Broelemann, D. Masip, G. Kasneci, T. Thiropanis, and S. Staab.

Explanation shift: Investigating interactions between models and shifting data distributions. *arXiv preprint arXiv:2303.08081*, 2023.

[70] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, pages 19–34, 2018.

[71] R. Paleja, M. Ghuy, N. Ranawaka Arachchige, R. Jensen, and M. Gombolay. The utility of explainable ai in ad hoc human-machine teaming. *Advances in Neural Information Processing Systems*, 34:610–623, 2021.

[72] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[73] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

[74] Y. Ramon, T. Vermeire, O. Toubia, D. Martens, and T. Evgeniou. Understanding consumer preferences for explanations generated by xai algorithms. *arXiv:2107.02624*, 2021.

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

[76] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535, 2018.

[77] S. Roy, L. J. Orr, and D. Suciu. Explaining query answers with explanation-ready databases. *Proc. VLDB Endow.*, 9(4):348–359, 2015.

[78] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *SIGMOD*, pages 1579–1590. ACM, 2014.

[79] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[80] C. Rudin and Y. Shaposhnik. Globally-consistent rule-based summary-explanations for machine learning models: Application to credit-risk evaluation. *J. Mach. Learn. Res.*, 24:16:1–16:44, 2023.

[81] C. Russell. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 20–28, 2019.

[82] B. Salimi and L. E. Bertossi. From causes for database queries to repairs and model-based diagnosis and back. In *ICDT*, 2015.

[83] B. Salimi and L. E. Bertossi. Query-answer causality in databases: Abductive diagnosis and view updates. In *Proc. UAI'15 Causal Inference Workshop*, 2015.

[84] B. Salimi and L. E. Bertossi. Causes for query answers from databases, datalog abduction and view-updates: The presence of integrity constraints. In *FLAIRS*, 2016.

[85] B. Salimi, L. E. Bertossi, D. Suciu, and G. V. den Broeck. Quantifying causal effects on query answering in databases. In *TaPP*, 2016.

[86] B. Salimi, J. Gehrke, and D. Suciu. Bias in OLAP queries: Detection, explanation, and removal. In *SIGMOD*, pages 1021–1035. ACM, 2018.

[87] P. Schmidt and A. D. Witte. *Predicting recidivism in north carolina, 1978 and 1980.* Inter-university Consortium for Political and Social Research, 1988.

[88] A. Selbst and J. Powles. "meaningful information" and the right to explanation. In *conference on fairness, accountability and transparency*, pages 48–48. PMLR, 2018.

[89] A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111, 2018.

[90] A. A. Shrotri, N. Narodytska, A. Ignatiev, K. S. Meel, J. Marques-Silva, and M. Y. Vardi. Constraint-driven explanations for black-box ML models. In *AAAI*, pages 8304–8314. AAAI Press, 2022.

[91] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods. In *AIES*, pages 180–186. ACM, 2020.

[92] K. Sokol and P. Flach. Explainability fact sheets: a framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 56–67, 2020.

[93] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

[94] K. S. Tai, V. Sharan, P. Bailis, and G. Valiant. Sketching linear classifiers over data streams. In *SIGMOD*, pages 757–772, 2018.

[95] T. Teofili, D. Firmani, N. Koudas, V. Martello, P. Merialdo, and D. Srivastava. Effective explanations for entity resolution models. In *ICDE*, pages 2709–2721, 2022.

[96] M. Tsang, S. Rambhatla, and Y. Liu. How does this interaction affect me? interpretable attribution for feature interactions. In *NeurIPS*, 2020.

[97] C. Umans. The minimum equivalent DNF problem and shortest implicants. *J. Comput. Syst. Sci.*, 63(4):597–611, 2001.

[98] D. Wang, Q. Yang, A. Abdul, and B. Y. Lim. Designing theory-driven user-centric explainable ai. In *CHI*, pages 1–15, 2019.

[99] H. Wang, N. Wang, and D. Yeung. Collaborative deep learning for recommender systems. In *KDD*, pages 1235–1244. ACM, 2015.

[100] D. S. Weld and G. Bansal. The challenge of crafting intelligible intelligence. *Communications of the ACM*, 62(6):70–79, 2019.

[101] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *Proc. VLDB Endow.*, 6(8):553–564, 2013.

[102] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *NLPCC*, volume 11839, pages 563–574. Springer, 2019.

APPENDIX A: PROOFS

Proof of Theorem 1

To see the decision version of Mrkp (also denoted as Mrkp) is NP-complete, we first need to show that Mrkp is in NP. We give an algorithm working for Mrkp as follows: (a) it first guesses a subset E of features of $\mathcal{X}$ for x; (b) it then checks whether E is a valid α-conformant relative key for x relative to context **I**. Here step (b) is in Ptime since the process of checking whether E is α-conformant is polynomial *w.r.t.* $|\mathbf{I}|$ and n, where n is the number of features of instance in **I** and $|\mathbf{I}|$ is the number of instances in **I**. The hardness of Mrkp is proved below.

We show that Mrkp is NP-hard by reduction from the minimum set cover (Msc) problem, which is known NP-complete [17]. Given an instance of Msc, *i.e.,* a universe U of elements $e_1, \ldots, e_m$, a collection $\mathcal{F}$ of subsets $S_1, \ldots, S_n$ of U, and an integer k, Msc is to decide whether there exist a k-cover, *i.e.,* k sets in $\mathcal{F}$ that together cover all m elements of U.

Given the Msc instance containing U, $\mathcal{F}$ and k, we construct a context **I** over n features $A_1, \ldots, A_n$ that consists of $m+1$ instances $\mathrm{x}_1, \ldots, \mathrm{x}_m$ and x. We show that there exists a k-minimum (*i.e.,* k features) 1-conformant key for x relative to **I** *if and only if* the Msc instance has a k-cover.

- $\mathrm{x} = (a_1, \ldots, a_n)$;
- for each $e_i \in U$ and $S_j \in \mathcal{F}$, if $e_i \in S_j$, then $\mathrm{x}_i[A_j] \neq a_j$;
- for any $\mathrm{x}_i[A_j]$ not decided above, set it to a distinct constant;
- the labels of all $m+1$ instances of **I** are distinct.

($\Rightarrow$) Assume k features $A_1, \ldots, A_k$ is the k-minimum 1-conformant key for x relative to **I**. Based on the construction process, we know for each $e_i \in U$ and S_j ($j \in [1,k]$), if $e_i \in S_j$, then $\mathrm{x}_i[A_j] \neq a_j$. Without loss of generality, assume that the union of $S_1, \ldots, S_k$ corresponding to $A_1, \ldots, A_k$ does not cover an element e_i in U, *i.e.,* e_i does not belong to any $S_1, \ldots, S_k$. This means that for each A_j ($j \in [1,k]$), $\mathrm{x}_i[A_j] = a_j$. That is, x_i and x have the same values in all features $A_1, \ldots, A_k$. We also know that the labels of x and x_i must be different. This implies a situation where x_i is identical to x in $A_1, \ldots, A_k$ but has a different label, which contradicts the fact that $A_1, \ldots, A_k$ is the k-minimum 1-conformant key for x relative to **I**. Hence, $S_1, \ldots, S_k$ is the k-cover.

($\Leftarrow$) Assume there exists a k-cover, *i.e.,* $S_1, \ldots, S_k$, together cover all m elements $e_1, \ldots, e_m$ of U. We know that the labels of all $m+1$ instances of **I** are distinct, we can map the elements $e_1, \ldots, e_m$ in U to the labels of $\mathrm{x}_1, \ldots, \mathrm{x}_m$ respectively. According to the construction, we know that for each S_j ($j \in [1,k]$), if $e_i \in S_j$, then $\mathrm{x}_i[A_j] \neq a_j$. We iteratively perform the following operation: for each $S_j (j \in [1,k])$, remove the elements that are in S_j from U. Since $S_1, \ldots, S_k$ is the k-cover, U must be empty. This means that there is no instance in $\mathrm{x}_1, \ldots, \mathrm{x}_m$ with the same values as x in k features $A_1, \ldots, A_k$ and also with different labels. That is, $A_1, \ldots, A_k$ uniquely determine the label of x and is naturally the k-minimum 1-conformant key for x relative to **I**.

Proof of Theorem 2(1)

To prove that there exists no Ptime algorithm for Mrkp that returns a $(1 - o(1)) \cdot \ln \alpha|\mathbf{I}|$-bounded relative key when $\alpha = 1$, we first introduce L-reduction [72], a specific type of approximation-preserving reduction.

L-reduction. Given two optimization problems P and P', an L-reduction from P to P' is a pair of polynomial functions (f, g), with two positive constants α and β, such that (a) f maps each instance x of P with optimal cost $\mathrm{OPT}_P(x)$ to an instance $f(x)$ of P' with optimal cost $\mathrm{OPT}_{P'}(f(x))$ satisfying $\mathrm{OPT}_{P'}(f(x)) \leq \alpha \cdot \mathrm{OPT}_P(x)$; and (b) g maps each solution y of P' to a solution $g(y)$ of P such that, if y is a solution of $f(x)$ for P', then $g(y)$ is a solution of x for P that satisfies $|\mathrm{OPT}_P(x) - \mathrm{cost}_P(g(y))| \leq \beta \cdot |\mathrm{OPT}_{P'}(f(x)) - \mathrm{cost}_{P'}(y)|$, where $\mathrm{cost}_{P'}(y)$ (resp. $\mathrm{cost}_P(g(y))$) denotes the cost of solution y (resp. $g(y)$) of problem P' (resp. P). If P' has a c-approximation ($c \geq 1$) algorithm, then the reduction naturally gives us an $\alpha \cdot \beta \cdot c$-approximation algorithm for P.

We then describe an L-reduction from Mrkp to the minimum set cover (Msc) problem. Recall that an instance of Msc consists of a universe U of elements and a collection $\mathcal{F}$ of subsets of U with $\bigcup_{S \in \mathcal{F}} S = U$, and it is to select a set $C \subseteq \mathcal{F}$ of minimum cardinality that covers U, *i.e.,* $\bigcup_{S \in C} S = U$. With this, we next give the specific L-reduction, namely functions f and g with constants α and β. The idea is to "reverse" the reduction in the proof of Theorem 1. More specifically, consider an instance of Mrkp, *i.e.,* inference context **I** and an instance x in **I** to be explained. Let x be $(a_1, \ldots, a_n)$. Note that $\alpha = 1$.

(1) Function f constructs an instance of Msc, *i.e.,* a universe U and a set $\mathcal{F}$ of subsets of U as follows: (a) U consists of m elements $e_1, \ldots, e_m$; (b) $\mathcal{F}$ consists of n sets $S_1, \ldots, S_n$, such that $e_i \in S_j$ if and only if $\mathrm{x}_i[A_j] \neq a_j$, for any $i \in [1,m]$ and $j \in [1,n]$; and (c) the labels of all m instances of **I** are distinct, and also all are different from that of x.

(2) For a cover $C \subseteq \mathcal{F}$ of U, function g maps it to the 1-conformant key E for x relative to **I** such that $A_i (i \in [1,n])$ is in E if and only if $S_i \in C$.

(3) $\alpha = \beta = 1$.

The above functions f and g with $\alpha = \beta = 1$ has already formed an L-reduction from Mrkp to the Msc variant. Since it is known that Msc cannot be approximated within an approximation factor of $(1 - o(1)) \cdot \ln m$ in polynomial time [22, 54], so does Mrkp.

Proof of Lemma 3

We prove that for any conformity bound $\alpha \in (0, 1]$, algorithm SRK always returns an α-conformant $\ln(\alpha|\mathbf{I}|)$-bounded key for x relative to **I**. Denote by OPT the offline optimal algorithm which always gives the key with minimal feature number. Denote by k_{SRK} and k_{OPT} are the feature number of keys given by SRK and OPT, respectively. With this, we want to prove for instance x with any **I**, $k_{\mathsf{SRK}} \leq k_{\mathsf{OPT}}(\ln(\alpha|\mathbf{I}|)$ where $|\mathbf{I}|$ represents the number of instances in **I**. We just need to consider the worst case, *i.e.,* the predictions of all instances in **I** are distinct. In this case, we should evaluate all other $|\mathbf{I}|$ instances in $|\mathbf{I}|$ for computing the key for x relative to **I**.

We next give the proof in two cases, *i.e.,* $\alpha = 1$ and $0 < \alpha < 1$.

• CASE 1: $\alpha = 1$. When producing an absolutely conformant explanation, we know that the eventual produced key should be satisfied over all other $|\mathbf{I}|$ instances. Let u_j be the number of instances that are consistent with the current key (*i.e.,* sharing identical values as x for the features composing the key) yet exhibit different predictions at iteration j of the while loop in SRK. It is known that the optimal key with k_{OPT} features in OPT definitely holds over the entire **I**, and naturally includes those u_j instances. Therefore, there must exist at least one feature in OPT that processes at least u_j/k_{OPT} instances. This means that in the next iteration in SRK, adding an feature to the key will remove at least u_j/k_{OPT} instances from u_j instances that do not satisfy the key. This is because SRK always picks the feature whose marginal contribution is largest at each iteration. That is, the selected feature proceeds the most number of instances that share the same value with x on the current key but with different predictions. With this, we know there are at most $u_j - u_j/k_{\mathsf{OPT}}$ instances left after iteration j. Putting all the iterations together, we have:

$$u_{j+1} \leq u_j - \frac{u_j}{k_{\mathsf{OPT}}} = (1 - \frac{1}{k_{\mathsf{OPT}}})u_j \leq (1 - \frac{1}{k_{\mathsf{OPT}}})(1 - \frac{1}{k_{\mathsf{OPT}}})u_{j-1} \leq \ldots \leq (1 - \frac{1}{k_{\mathsf{OPT}}})^{j+1}u_0,$$

where $u_0 = |\mathbf{I}|$ due to the fact that there are exactly $|\mathbf{I}|$ instances in **I** at the beginning. We know that SRK will be done once we get the iteration i such that $u_i \leq 1$ since it implies that we need to select at most one more features. The key point is to show how can we compute the i. That is, how large does i need to be to guarantee that $u_i \leq 1$.

By the inequality $1 - x \leq e^{-x}$, we have:

$$(1 - \frac{1}{k_{\mathsf{OPT}}})^i \leq e^{-\frac{i}{k_{\mathsf{OPT}}}}.$$

With this, we have:

$$|\mathbf{I}|e^{-\frac{i}{k_{\mathsf{OPT}}}} \leq 1 \Leftrightarrow e^{-\frac{i}{k_{\mathsf{OPT}}}} \leq |\mathbf{I}|^{-1} \Leftrightarrow -\frac{i}{k_{\mathsf{OPT}}} \leq -\ln|\mathbf{I}| \Leftrightarrow i \geq k_{\mathsf{OPT}} \ln|\mathbf{I}|,$$

which shows that after $k_{\mathsf{OPT}} \ln|\mathbf{I}|$ steps the remaining number of instances u_i is smaller or equal to 1. Thus, when $\alpha = 1$, $k_{\mathsf{SRK}} \leq \ln|\mathbf{I}|k_{\mathsf{OPT}}$.

• CASE 2: $0 < \alpha < 1$. In this case, the eventual key only needs to be held on a fraction α of the instances of **I**. This implies that the straightforward performance analysis of SRK becomes more complicated since OPT may hold over a different set of instances from SRK. Let G_{SRK} (resp. G_{OPT}) be the number of instances that satisfy the dependency condition of relative keys over the features picked by SRK (resp. OPT). To analyze the performance of SRK, we first give a following lemma.

Lemma A: For $k_{\mathsf{SRK}} \geq k_{\mathsf{OPT}} \geq 1$, there will be

$$G_{\mathsf{SRK}} \geq \left(1 - \left(1 - \frac{1}{k_{\mathsf{OPT}}}\right)^{k_{\mathsf{SRK}}}\right)G_{\mathsf{OPT}}.$$

□

Proof: Initially, the key E is empty, and with each iteration of SRK, one feature is added to E. When the selected feature computed by the j-th iteration of SRK is incorporated into E, let g_j denote the number of newly added instances with values that are not entirely identical to all the features constituting the current E. Clearly, $G_{\mathsf{SRK}} = \sum_{j=1}^{k_{\mathsf{SRK}}} g_j$. Based on the pigeonhole principle, for the first iteration, $g_1 \geq \frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}}$ holds true. This also applies to subsequent iterations, with g_2 being greater than or equal to $\frac{G_{\mathsf{OPT}} - g_1}{k_{\mathsf{OPT}}}$. Combining the n iterations, we obtain:

$$g_{n+1} \geq \frac{G_{\mathsf{OPT}} - G_n}{k_{\mathsf{OPT}}} \tag{1}$$

We define a sequence. Specifically, let $s_1 = \frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}}$ and generally $s_n = (1 - \frac{1}{k_{\mathsf{OPT}}})s_{n-1}$. Hence, $s_n = (1 - \frac{1}{k_{\mathsf{OPT}}})^{n-1}\frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}}$.

We then prove $G_{n+1} \geq \sum_{i=1}^{n+1} s_i$ by induction on the n iterations.

Basis. It is known that $G_1 = g_1 \geq \frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}}$.

Induction hypothesis. Assume $G_n \geq \sum_{i=1}^{n} s_i$.

Induction step. With inequality (1) and the induction hypothesis, we have

$$G_{n+1} = G_n + g_{n+1} \geq G_n + \frac{G_{\mathsf{OPT}} - G_n}{k_{\mathsf{OPT}}} = \frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}} + (1 - \frac{1}{k_{\mathsf{OPT}}})G_n$$

$$\geq \frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}} + (1 - \frac{1}{k_{\mathsf{OPT}}})\sum_{i=1}^{n} s_i = \frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}} + \sum_{i=1}^{n} s_{i+1} = s_1 + \sum_{i=2}^{n+1} s_i = \sum_{i=1}^{n+1} s_i.$$

The induction completes. With this and the sequence above, we have

$$G_{\mathsf{SRK}} \geq \sum_{i=1}^{k_{\mathsf{SRK}}} s_i = \sum_{i=1}^{k_{\mathsf{SRK}}} \left(1 - \frac{1}{k_{\mathsf{OPT}}}\right)^{i-1} \frac{G_{\mathsf{OPT}}}{k_{\mathsf{OPT}}} = \left(1 - \left(1 - \frac{1}{k_{\mathsf{OPT}}}\right)^{k_{\mathsf{SRK}}}\right) G_{\mathsf{OPT}}.$$

The lemma holds. □

We continue to prove the theorem. For the ease of presentation, we let $p = \lceil \alpha |\mathbf{I}| \rceil$. We only need to consider the case $k_{\mathsf{OPT}} > 1$ since SRK could also easily find the same result with OPT when $k_{\mathsf{OPT}} = 1$. We assume there is another solution k'_{SRK} produced by SRK. We let the least value of k'_{SRK}, denoted by r, such that $G_{\mathsf{SRK}} = p - c$ for a positive constant c. Note that we also should ensure $r \geq k_{\mathsf{OPT}}$. By Lemma A and $G_{\mathsf{OPT}} \geq p$, we actually have below inequalities:

$$\begin{aligned} p - c \geq \left(1 - \left(1 - \frac{1}{k_{\mathsf{OPT}}}\right)^r\right) G_{\mathsf{OPT}} \geq \left(1 - \left(1 - \frac{1}{k_{\mathsf{OPT}}}\right)^r\right) p \Rightarrow \\ -\left(1 - \frac{1}{k_{\mathsf{OPT}}}\right)^r \leq \frac{-c}{p} \Rightarrow \\ -r \ln\left(1 - \frac{1}{k_{\mathsf{OPT}}}\right) \leq \ln p - \ln c. \end{aligned} \tag{2}$$

With the inequality $\ln(1 + x) \leq x$, equation (2) becomes

$$\frac{r}{k_{\mathsf{OPT}}} \leq \ln p - \ln c. \tag{3}$$

According to the SRK process, the instances where those k'_{SRK} holds true are definitely a subset of those for k_{SRK}. Hence, we know the r features (*i.e.*,, the lease value of k'_{SRK}) exactly have contained $p - c$ satisfied instances so far. This implies that only at most c more features are needed for p instances to form the key, as each feature contains at least one potential instance. Thus, adding the possible c features to the solution of SRK, now we have:

$$k_{\mathsf{SRK}} \leq r + c. \tag{4}$$

With inequality (3) and (4), we have

$$\frac{k_{\mathsf{SRK}}}{k_{\mathsf{OPT}}} \leq \frac{r + c}{k_{\mathsf{OPT}}} \leq \ln p - \ln c + \frac{c}{k_{\mathsf{OPT}}}. \tag{5}$$

The right-hand side of inequality (5) obtains its minimum value when $c = k_{\mathsf{OPT}}$, which further shows

$$\frac{k_{\mathsf{SRK}}}{k_{\mathsf{OPT}}} \leq \ln p - \ln k_{\mathsf{OPT}} + 1. \tag{6}$$

$1 - \ln k_{\mathsf{OPT}}$ is less than 0 when $k_{\mathsf{OPT}} > e$, otherwise $1 - \ln k_{\mathsf{OPT}}$ is negligible which can be omitted since k_{OPT} can only be 1 or 2. Hence, we have

$$\frac{k_{\mathsf{SRK}}}{k_{\mathsf{OPT}}} \leq \ln p = \ln \lceil \alpha |\mathbf{I}| \rceil.$$

Putting the two cases together, the theorem holds.

Proof of Theorem 4

We construct a counter-example to verify that there exists no deterministic algorithm for ORKM that is $O(n)$-competitive when the conformity bound α is fixed to 1. We first assume *w.l.o.g.* initially $\mathbf{I}$ and E are empty and all online instances have predictions different from that of x_0. Let the number of online instances be n which is equal to the total number of features. Consider there is an adversary that controls the arrival order and features of those n online instances. Let $\mathcal{A}$ be any deterministic online algorithm for ORKM. Upon the arrival of each x_i, the adversary can always construct the next instance x_{i+1} so that it is identical to x_0 on all those features picked by $\mathcal{A}$ for E_i, while differing on all the other features. That is, the adversary can always ensure that the key currently selected by $\mathcal{A}$ becomes invalid when the next instance arrives. In contrast, consider an offline algorithm OFF that just picks the one feature for which all n instances have distinct values. Then after processing all n instances, $\mathcal{A}$ already picks n features for E_n (*i.e.,* $\mathsf{succinct}(E_n) = n$) while the succinctness of E_i returned by OFF for all $i \in [1, n]$ is always 1. By the definition of competitiveness, there exists no deterministic algorithm for ORKM that is $O(n)$-competitive when the conformity bound α is fixed to 1.

Proof of Theorem 5

Assume *w.l.o.g.* we have an offline optimal algorithm, denoted by OPT, that always returns the most succinct keys relative to $\{\mathrm{x}_1, \ldots, \mathrm{x}_t\}$ for each x_t, using full knowledge of the entire $\mathbf{I}$ known in advance. Let k_{OSRK} and k_{OPT}, respectively, be the succinctness of keys computed by OSRK and OPT over time.

We first demonstrate that OSRK is feasible to output 1-conformant key. This is because, when $\alpha = 1$ for each arriving instance x_t with a different prediction with x_0, either E_{t-1} is already the valid 1-conformant key, or OSRK definitively selects a feature to expand the key E_t. We next observe that there are three stages in OSRK where features can be added to E_t while serving $\mathrm{x}_1, \ldots, \mathrm{x}_t$: (1) when handling the first instance that predicts differently from x_0 (line 6); (2) before weight augmentation (line 11), that is, the instance weight is greater than the threshold $\log p_t$ where p_t is the number of instances with different predictions encountered so far, and (3) during weight augmentation (line

15). Denote by random variables X, Y and Z the number of features added to E_t in these three stages, respectively. It then becomes evident that the number of features in the key generated by OSRK is the sum of the expectations of X, Y and Z. That is, $k_{\mathrm{OSRK}} = \mathbb{E}(X) + \mathbb{E}(Y) + \mathbb{E}(Z)$. We next analyze the expectations of X, Y and Z in sequence.

• Computing $\mathbb{E}(X)$. Since each feature weight w_i is initialized to be $\frac{1}{n}$ at most, clearly we have

$$\mathbb{E}(X) = \sum_{i \in \mathbf{A}} w_i \leq \frac{1}{n} * n = 1.$$

Note $\mathbf{A} = \{1, \ldots, n\}$ where i corresponds to feature A_i in feature space.

• Computing $\mathbb{E}(Y)$. We consider the case when a feature is added to E_{t-1} before the weight augmentation stage while processing the online instance x_t. We *w.l.o.g.* consider the worst case, *i.e.,* each online x_t has different prediction with x_0. This implies $\mu_i > \log i$ for $i \in [0, \ldots, t]$ since $i = p_i$. We know this stage occurs only when the feature has never been selected to be added to E_{t-1} before. Otherwise, E_{t-1} is must be a valid 1-conformant key. Since w_j is the probability of feature A_j being selected and S_i represents the set of features where x_i and x_0 have different values for each $i \in [0, \ldots, t]$, we have

$$\mathbb{E}(Y) \leq \sum_{i=1}^{t} \prod_{j \in S_i} (1 - w_j) \leq \sum_{i=1}^{t} (1 - \frac{\mu_i}{|S_i|})^{|S_i|} \leq \sum_{i=1}^{t} e^{-\mu_i} < \sum_{i=1}^{t} e^{-\log i} < \sum_{i=1}^{t} e^{-\ln i} = \sum_{i=1}^{t} \frac{1}{i} < \ln t + 1.$$

• Computing $\mathbb{E}(Z)$. In order to achieve this, there are two critical aspects to consider. Firstly, we need to compute the number of weight augmentation while processing t instances. Secondly, it is essential to determine the features number added to the key during each weight augmentation. Denote by N the total number of weight augmentation to handle all the t instances.

We demonstrate that $N \leq \log n * k_{\mathrm{OPT}}$. This is due to the following reasons: (1) for each feature A_i, its weight is initialized as $w_i = 2^{-k}$ where k is the maximum integer such that $2^{-k} < \frac{1}{n}$. w_i cannot be greater than 2, because when $w_i > 1$, feature A_i must already be part of the key. Simultaneously, we are aware that in each weight augmentation, w_i doubles. Based on this, after at most $\log n$ times, the initial weight of w_i will increase to 1; (2) there has to be at least one feature belonging to OPT participating in each weight augmentation; (3) for each feature in OPT, there may be at most $\log n$ times of weight augmentation; (4) after $\log n$ times, the key generated by OSRK already fully encompasses the result of the key generated by OPT.

We know only if $\mu_t \leq \log t$ (worst case), the weight augmentation can be performed. Therefore, it is easy to know that the expected number of features added during the run of weight augmentation is $\sum_{i \in S_t} w_i = \mu_t \leq \log t$. Hence, we have

$$\mathbb{E}(Z) \leq \log t * \log n * k_{\mathrm{OPT}} = O(\log t \log n) k_{\mathrm{OPT}}.$$

Combining $\mathbb{E}(X)$, $\mathbb{E}(Y)$ and $\mathbb{E}(Z)$, the theorem holds.

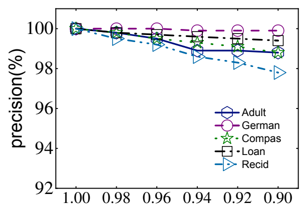
Proof of Theorem 6

We prove that for each x_t, SSRK returns a $(\log m \cdot \log n)$-bounded key for x_0 relative to $\mathbf{I} = \{x_1, \ldots, \mathsf{x}_t\}$ when $\alpha = 1$. Denote by OPT the offline optimal algorithm that always returns the most succinct keys for each x_t, with the entire sequence of $\mathbf{I}$ known in advance. Let k_{SSRK} and k_{OPT}, are the succinctness of keys computed by SSRK and OPT over time, respectively.

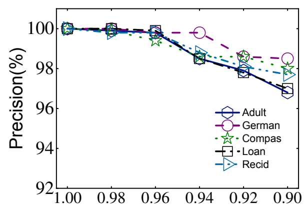
We first show SSRK can always output 1-conformant key. That is, for any online instance x_t with different prediction with x_0, if the current E_{t-1} does not satisfy 1-conformant key, SSRK always picks at least a feature to add to key E_t such that E_t is valid. This is because SSRK ensures that (1) features are added to E_t solely during the iterations of the weight augmentation stage; (2) the updated value of the potential function after each iteration is consistently no greater than its previous value, achieved by continuously adding features to E_t; (3) prior to the subsequent iteration, the new value of the potential function must surpass the value after the preceding iteration, guaranteeing a seamless progression to the next iteration. With this, to compute k_{SSRK}, we only need to determine the number of times weight augmentation occurs and the number of features added to E_t in a single iteration during each weight augmentation for x_t, denoted as X and Y respectively. That is, $k_{\mathrm{SSRK}} = X * Y$.

• Computing X. In the worst case, all instances in $\mathbf{I}$ have predictions different from x_0. We know $m = |\mathbf{I}|$. It is important to note that weight augmentation occurs only when the previous key E_{t-1} is not the valid 1-performance key. We argue that if μ_t is greater than or equal to 1, E_{t-1} is already the valid one. That is, when weight augmentation cannot be performed, μ_t is at most 1. This can be deduced from the following observations: (1) the initial value of the potential function Φ is less than m^2, as each feature set $w_i = \frac{1}{2n}$ initially; (2) Φ is non-increasing throughout the iterations; (3) if $\mu_t \geq 1$, $\Phi \geq m^{2\mu_t} \geq m^2$, which contradicts (1) and (2). By definition, when μ_t is at most 1, every feature weight w_i must also be less than or equal to 1. We know that for each w_i, the initial value is $\frac{1}{2n}$, and the weight doubles at least each time, so it takes at most $\log 2n$ iterations to reach 1. This also implies that for each feature in OPT, it may undergo at most $\log 2n$ weight augmentation operations. We also know when performing weight augmentation to process the online instance x_t, at least one feature in OPT is involved. Hence, in the worst case, after $\log 2n$ times of weight augmentation, the key produced by SSRK has already covered the results of the key generated by OPT. As a result, we have $X \leq O(\log n) k_{\mathrm{OPT}}$.

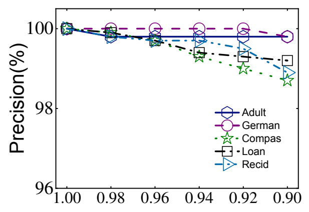
• Computing Y. Consider the key E_{t-1} that does not satisfy the 1-conformant key for online instance x_t, which implies that weight augmentation must be performed. Let δ_i represent the weighted increment for the weight w_i of feature A_i (*i.e.,* feature index i belonging to S_t). Naturally, $\Delta_t = \sum_{i \in S_t} \delta_i$. Recall that $m = |\mathbf{I}|$. Before weighting μ_t, the contribution of x_t to the potential function is $m^{2\mu_t}$. We know



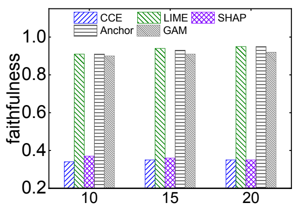
(a) SRK: precision with varying α



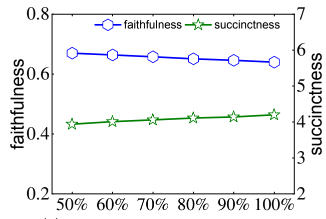
(b) OSRK: precision with varying α



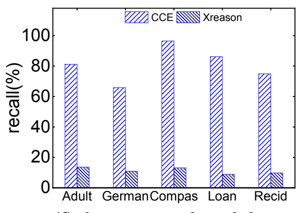
(c) SSRK: precision with varying α



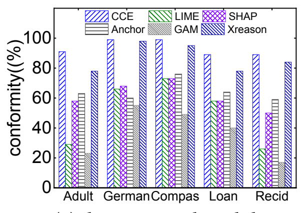
(d) varying #-bucket



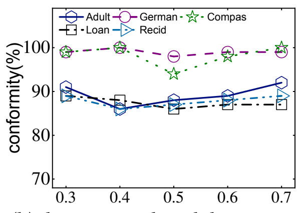
(e) SRK: varying context size



(f) dynamic explainability



(g) dynamic explainability



(h) dynamic explainability: vary $\Delta\mathbf{I}$

Figure 4: The additional quality and benefits of relative keys

SSRK must ensure that the value of the potential function decreases monotonically. This implies that, by continuously performing iterations of adding features to E_t, the contribution of x_t to the potential function must not exceed $m^{2\mu_t}$. We argue that adding at most $4\log m$ features can satisfy this requirement. This is due to the following reasons: (1) for any feature index i belonging to S_t, we can set the probability of it not being selected as $1-\frac{\delta_i}{2}$, assuming all of feature A_i are arranged in an uniform distribution. Here, $1-\frac{\delta_i}{2}\leq 1$ because $\frac{\Delta_t}{2}\leq 1$; (2) according to the definition of the potential function, there will be a contribution to the potential function only when all the features in S_t have not been selected. Thus, after $4\log m$ selections, $(1-\sum_{i\in S_t}\frac{\delta_i}{2})^{4\log m}=(1-\frac{\Delta_t}{2})^{4\log m}\leq m^{-2\Delta_t}$. Consequently, after $4\log m$ selections, the maximum contribution of instance x_t to the potential function is given by $m^{-2\Delta_t}m^{2(\mu_t+\Delta_t)}=m^{2\mu_t}$, which corresponds to the contribution of t to the potential function prior to performing the iteration. With all, we know SSRK picks at most $4\log m$ features to E_t in each weight augmentation.

Combining X and Y, we have

$$k_{\text{SSRK}}=X*Y\leq(\log m\log n)k_{\text{OPT}},$$

from which the theorem holds.

APPENDIX B: ADDITIONAL EXPERIMENTS

This section serves as a supplement to the experimental section. In addition to providing comprehensive experimental results for Section 7.3, we further evaluated the capability of CCE in explaining dynamic models. All the results are run with the same configurations as in Section 7.

Exp-1: Flexible trade-offs. We assessed the effectiveness of α-conformant relative keys in terms of precision. Figures 4a-4c respectively depict the precision of SRK, OSRK and SSRK as the conformity bound α varies from 1 to 0.9, and tell us the following. (a) it is reasonable that precision over all datasets shows a steady decrease as α decreases. However, the decline is rather minor. For instance, in the batch model over Compas the precision of SRK only decreased by 0.1% and 1.0% when varying α from 1 to 0.98 and 0.9, respectively. Similarly for OSRK and SSRK and other datasets. This verifies that our algorithms are robust against varying α. (b) On the other hand, the precision is significantly higher than the baselines (*i.e.,* the worst precision in theory) of all the α over all the datasets. When α is set to 0.9 in the batch model, the precision of Adult, German, Compas, Loan and Recid are 99.3%, 100%, 99.0%, 99.4% and 98.3% respectively, which are notably better than the theoretical value, *i.e.,* 90%. The results are consistent with OSRK and SSRK. (3) Moreover, the benefits of our algorithms are even more evident when α is smaller. On average, over the three algorithms the gap between our algorithms and the baselines over all the dataset increase from 1.9% to 9.3% when α is varied from 0.98 to 0.9. The results demonstrates that the lower the criteria, the better our algorithms perform than the expected.

Exp-2: Impact of numerical features. Fig 4d shows the impact of bucketing for numerical features on the faithfulness of explanations over Adult. Consistent with Section 7.3, CCE consistently outperforms other methods across different #-bucket. Over all #-bucket, on average CCE is 62.9%, 3.6%, 62.7%, 61.7% than LIME, SHAP, Anchor and GAM. This demonstrates that the capability of CCE to explain unseen instance remains robust to the #-bucket for the numerical features.

Exp-3: Impact of context. Varying the context size $|\mathbf{I}|$ from 50% to 100% over Adult, we assessed the faithfulness and succinctness of CCE (SSRK). As illustrated in Fig 4e, with an increase in $|\mathbf{I}|$, the faithfulness of CCE decreases while its succinctness increases. This is expected, as an increase in the context size $\mathbf{I}$ on which the relative key depends would naturally lead to increased succinctness. Perturbing more features also makes it more likely to change the prediction results of new instances.

Exp-4: Dynamic explainability. We next evaluated the capability of CCE in explaining dynamic models that change over time during model inference.

Dynamic explainability ability. CCE supports the explanation of dynamic models that evolve on-the-fly at inference time, with or without knowing when the model M changes. When CCE knows exactly when M changes, it naturally cleans its context and switches to inference instances and predictions collected from the updated M.

The more tricky scenario is when the model M changes without notifying the client, *i.e.,* CCE does not know exactly when M evolves. In such a case, CCE uses a sliding window method to dynamically adjust context $\mathbf{I}$ for explanation, where in each step it shifts $\mathbf{I}$ with a pre-configured $\Delta\mathbf{I}$ number of new instances, and accordingly, drops $\Delta\mathbf{I}$ "oldest" instances. Normally $\Delta\mathbf{I}$ is smaller than $\mathbf{I}$ so that explanations from CCE can reflect changes of M timely.

There can be a situation where as an instance x appears in multiple overlapping contexts collected in consecutive steps; hence, it can have multiple different explanations (keys) relative to the contexts. To this end, CCE implements several resolution policies: (a) First-wins: once the relative key for x is computed with an earlier context, it will not be updated for later context that also includes x. (b) Last-wins: the key for x will be relative to the latest context that contains x. (c) Union-key: it takes the union of all the keys for x relative to contexts that contains x. CCE uses the last-wins policy by default.

Experimental setup. To control the impact of the ML model and accurately evaluate the effectiveness of explanation methods, we created a 5-phase dynamic model as follows. We divided each dataset (both training and inference sets) into 5 parts of equal size, each trained with a distinct XGBoost model. This gives us a sequence of 5 models, simulating a dynamic model with 5 updates in a controlled way.

When the explanation method is aware of the changes to the ML model and is updated accordingly, the results are exactly the same as those in Section 7.3. Hence, below we focus on the scenario where the explanation methods are oblivious to model changes.

Quality of explanation. We first evaluated the quality of explanations computed by all methods. For each phase, we sampled 20 instances for explanation; we used the result of SRK with all inference instances of the phase as the reference, and compared the explanations of CCE (without knowing the inference set of each phase), LIME, SHAP, Anchor, GAM and Xreason with the reference to calculate the quality measures, *e.g.,* conformity and recall.

(a) Recall and succinctness. We find that the succinctness of explanations of all methods are similar to that of Section 7.3. In particular, the formal explanation returned by Xreason almost consumes half of the features of the explained instance, which makes it even less suitable for the dynamic setting as shown by the recall (Fig. 4f), *e.g.,* 13.6%, 10.9%, 13.1%, 8.9% and 9.8% over Adult, German, Compas, Loan and Recid, respectively, while the recall of CCE is 81.2%, 65.8%, 96.5%, 86.2% and 74.1%. Here the recall of Xreason is calculated by pairing with the reference explanation; similarly for CCE.

(b) Conformity. As shown in Fig. 4g, for dynamic models, CCE has the highest conformity among all methods over all datasets. Moreover, while all methods become less accurate than for static models in Section 7.3, CCE has the least decrease in conformity among all, *e.g.,* by 6.6% compared to 23.6%, 16.4%, 17.2%, 23.8% and 13.4% for LIME, SHAP, Anchor, GAM and Xreason, respectively. Xreason has better conformity than the other heuristic explanations since its succinctness is much worse, which makes very few instances matching its explanations. The results for precision are similar.

Impact of $\Delta\mathbf{I}$. We also evaluated the impact of parameter $\Delta\mathbf{I}$ of CCE for dynamic models, *i.e.,* the size of the "sliding" step of the context As shown in Fig. 4h, the performance, *e.g.,* conformity of CCE is robust against varying $\Delta\mathbf{I}$ over all datasets.