Adversarial Robustness for Tabular Data through Cost and Utility Awareness

Klim Kireev,[†] Bogdan Kulynych,[†] Carmela Troncoso

EPFL SPRING Lab

Abstract—Many safety-critical applications of machine learning, such as fraud or abuse detection, use data in tabular domains. Adversarial examples can be particularly damaging for these applications. Yet, existing works on adversarial robustness primarily focus on machine-learning models in image and text domains. We argue that, due to the differences between tabular data and images or text, existing threat models are not suitable for tabular domains. These models do not capture that the costs of an attack could be more significant than imperceptibility, or that the adversary could assign different values to the utility obtained from deploying different adversarial examples. We demonstrate that, due to these differences, the attack and defense methods used for images and text cannot be directly applied to tabular settings. We address these issues by proposing new cost and utility-aware threat models that are tailored to the adversarial capabilities and constraints of attackers targeting tabular domains. We introduce a framework that enables us to design attack and defense mechanisms that result in models protected against cost or utility-aware adversaries, for example, adversaries constrained by a certain financial budget. We show that our approach is effective on three datasets corresponding to applications for which adversarial examples can have economic and social implications.

I. INTRODUCTION

Adversarial examples are inputs deliberately crafted by an adversary to cause a classification mistake. They pose a threat in applications for which such mistakes can have a negative impact on deployed models (e.g., a financial loss [1] or a security breach [2–4]). Adversarial examples also have positive uses. For instance, they offer a means of redress in applications in which classification causes harm to its subjects (e.g., privacy-invasive applications [5–7]).

The literature on adversarial examples largely focuses on image [8–13] and text domains [14–19]. Yet, many of the applications where adversarial examples are most damaging or helpful are not images or text. High-stakes fraud and abuse detection systems [20], risk-scoring systems [1], operate on *tabular data*: A cocktail of categorical, ordinal, and numeric features. As opposed to images, each of these features has its own different semantics. For example, in a typical representation of an image, all dimensions of an input vector are similar in their semantics: they represent a color of a pixel. In tabular data, one dimension could correspond to a numeric

value of a person's salary, another to their age, and another to a categorical value representing their marital status.

The properties of the image domain have shaped the way adversarial examples and adversarial robustness are approached in the literature [11] and have greatly influenced adversarial robustness research in the text domain. In this paper, we argue that adversarial examples in tabular domains are of a different nature, and adversarial robustness has a different meaning. Thus, the definitions and techniques used to study these phenomena need to be revisited to reflect the tabular context.

We argue that two high-level differences need to be addressed. First, imperceptibility, which is the main requirement considered for image and text adversarial examples, is ill-defined and can be irrelevant for tabular data. Second, existing methods assume that all adversarial inputs have the same value for the adversary, whereas in tabular domains different adversarial examples can bring drastically different gains.

Imperceptibility and semantic similarity are not necessarily the primary constraints in tabular domains. The existing literature commonly formalizes the concept of "an example deliberately crafted to cause a misclassification" as a *natural example*, i.e., an example coming from the data distribution, that is *imperceptibly* modified by an adversary in a way that the classifier's decision changes. Typically, imperceptibility is formalized as closeness according to a mathematical distance such as L_p [21, 22].

In tabular data, however, imperceptibility is not necessarily relevant. Let us consider the following toy example of financial-fraud detection: Assume a fraud detector takes as input two features: (1) transaction `amount`, and (2) `device` from which the transaction was sent. The adversary aims to create a fraudulent financial transaction. The adversary starts with a natural example (`amount`=$200, `device`=Android phone) and changes the feature values until the detector no longer classifies the example as fraud.

In this example, *imperceptibility is not well-defined*. Is a modification to the `amount` feature from $200 to $201 imperceptible? What increase or decrease would we consider perceptible? The issue is even more apparent with categorical data, for which standard distances such as L_2, L_∞ cannot even capture imperceptibility: Is a change of the `device` feature from Android to an iPhone imperceptible?

Even if imperceptibility was well-defined, *imperceptibility might not be relevant*. Should we only be concerned about adversaries making "imperceptible" changes, e.g., modifying `amount` from $200 to $201? What about attack vectors in

† Contributed equally.

which the adversary evades detection while changing the transaction by a "perceptible" amount: from $200 to $2,000?

Formalizing adversarial examples as imperceptible modifications narrows the mathematical tools that can be used to study adversarial examples in their broad sense. In the case of tabular data, this prevents the study of techniques that adversaries could employ in "perceptible", yet effective ways.

We argue that in tabular data the primary constraint should be *adversarial cost*, rather than any notion of similarity. Instead of looking at how visually or semantically similar are the feature vectors, the focus should be on *how costly it is for an adversary to enact a modification*. Costs capture the effort of the adversary, e.g., financial or computational. "How much money does the adversary have to spend to evade the detector?" better captures the possibility that an adversary deploys an attack than establishing a threshold on the L_p distance the adversary could tolerate. In the fraud-detection example, regardless of whether a change from Android to iPhone is imperceptible and semantically similar or not, it is certain that the change costs the adversary a certain amount of resources. How significant are these costs determines the likelihood of the adversary deploying such an attack.

Different tabular adversarial examples are of different value to the adversary. In the literature, with a notable exception of Zhang and Evans [23], defenses against adversarial examples implicitly assume that all adversarial examples are equal in their importance [9, 13, 24–26]. In tabular data domains, however, different adversarial examples can bring very different *gains* to the adversary. In the fraud-detection example, if a fraudulent transaction with transaction `amount` of $2,000 successfully evades the detector, it could be significantly more profitable than a transaction with `amount` of $200.

Using the adversarial cost as the primary constraint for adversarial examples provides a natural way to incorporate the variability in adversarial gain. The adversary is expected to care about the profit obtained from the attack, i.e., the difference between the cost associated with crafting an adversarial example, and the gain from its successful deployment. We call this difference the *utility* of the attack. We show how utility can be incorporated into the design of attacks to ensure their economic profitability, and into the design of defenses to ensure protection against adversaries that focus on profit.

In this paper, we introduce a framework to build adversarial examples tailored to tabular data. Our contributions are:

- We propose two *adversarial objectives* for tabular data that address the limitations of the standard approaches: a *cost-bounded* objective that substitutes standard imperceptibility constraints with adversarial costs; and a novel *utility-bounded* objective in which the adversary adjusts their expenditure on different adversarial examples proportionally to the potential gains from deploying them.
- We propose a practical attack algorithm based on greedy best-first graph search for crafting adversarial examples that achieve the objectives above.

- We propose a new method for adversarial training to build classifiers that are practically resistant to adversaries pursuing our adversarial objectives.
- We empirically evaluate our attacks and defenses in realistic conditions demonstrating their applicability to real-world security scenarios. Our evaluation shows that cost-bounded defenses that ascribe equal importance to every example, as traditional approaches do, can degrade robustness against adversaries for whom some attacks have more value than others. Defenses crafted against these adversaries, however, perform well against both cost-oriented and utility-oriented adversaries.

II. EVASION ATTACKS

This section introduces the notation and formal setup of *evasion attacks* [see, e.g., 27, 28] in tabular domains.

Feature Space in Tabular Domains. The input domain's *feature space* $\mathbb{X}$ is composed of m features: $\mathbb{X} \subseteq \mathbb{X}_1 \times \mathbb{X}_2 \times \cdots \times \mathbb{X}_m$. For example $x \in \mathbb{X}$, we denote the value of its i-th feature as x_i. Features x_i can be categorical, ordinal, or numeric. Each example is associated with a binary class label $y \in \{0, 1\}$.

Target Classifier. We assume the adversary's *target* to be a binary classifier $F(x) \in \{0, 1\}$ that aims to predict the class y to which an example x belongs. It is parametrized by a decision function $f(x) \in [0, 1]$ such that $F(x) = \mathbf{1}[f(x) > {}^1\!/_2]$. The output of $f(x)$ can be interpreted as a score for x belonging to the positive class ($y = 1$). We focus on binary classification as it is the task in which adversarial dynamics typically arise in tabular domains (e.g., fraud detection [20] or risk-scoring systems [1]).

Adversarial Examples. An evasion attack proceeds as follows: The adversary starts with an initial example $x \in \mathbb{X}$ with a label $y = y_s$. We call this class the *adversary's source class*. The adversary's goal is to modify x to produce an *adversarial example* x^* that is classified as $F(x^*) = y_t$, $y_s \neq y_t$. We call this the *adversary's target class*. The attack is *successful* if the adversary can produce such an adversarial example. Depending on the adversarial objective, the adversarial example might also need to satisfy additional constraints, as detailed in Section III.

Because an attack is performed using an adversarial example, as in the literature, we use the terms *adversarial example* and *attack* interchangeably.

Our methods can be used in a multi-class setting as they are agnostic to which class is the target one. Our notation, however, is specific to the binary setting for clarity.

Adversarial Model. In terms of capabilities, we assume the adversary can only perform modifications that are within the domain constraints. In the fraud-detection example, the adversary can change the transaction amount, but the value must be positive. For a given initial labeled example (x, y), we denote the set of feasible adversarial examples that can be reached within the capabilities of the adversary as $\mathcal{F}(x, y) \subseteq \mathbb{X}$.

In terms of knowledge, we assume that the adversary has black-box access to the target classifier: The adversary can

issue queries using arbitrary examples and obtain $f(x)$. In our evaluation (Section VI) we compare this adversary against existing attacks with white-box access to the gradients.

Preservation of Semantics. It is common to require that an adversarial example is *semantics-preserving* [21, 29]: the adversarial example retains the same true class as the original example. We do not impose such a requirement. The only constraint we impose is that the modifications leading to an adversarial example *are feasible within the domain constraints*, i.e., that the adversarial example belongs to the set $\mathcal{F}(x, y)$. This is because in tabular domains limiting $\mathcal{F}(x, y)$ to those adversarial examples that also preserve semantics is counter-productive: As long as the adversary successfully achieves their goal with an adversarial example that is feasible given their budget (see Section III), the attack presents a valid threat.

III. ADVERSARIAL OBJECTIVES IN TABULAR DATA

As we detail in Section I, the approaches to adversarial modeling tailored to image or text data have two critical limitations when applied to tabular domains:

1. *Focus on imperceptibility and semantic similarity.* Neither closeness to natural examples in L_p distance, nor closeness in terms of semantic similarity, is a well-applicable definition of adversarial examples in tabular domains. This is because such similarities are either ill-defined for mixed-type features (e.g., as is the case with L_p distance), or potentially irrelevant to the quantification of adversarial constraints (both L_p and semantic similarity).

2. *Assuming all adversarial examples are equally useful.* Most existing defenses against adversarial examples do not distinguish different attacks in terms of their value for the adversary. In tabular domains, due to the inherent heterogeneity of the data, some attacks could bring significantly more gain to the adversary.

Next, we propose adversarial objectives which aim to address these limitations.

A. Cost-Bounded Objective

Evasion attacks using adversarial costs were first formalized in early works on adversarial machine learning [30, 31]. In these works, the adversary aims to find evading examples with minimal cost. Since the discovery of adversarial examples in computer vision models [8], this formalization was largely abandoned in favor of constraints based on L_p and other mathematical distances (e.g., Wasserstein distance [32] or LPIPS [33][34]). In this work, we revisit the cost-oriented approach, which better reflects the adversary's capabilities in tabular domains.

In a standard way to obtain an adversarial example [13], the adversary aims to construct an example that maximizes the classification loss $\ell(\cdot, \cdot)$ incurred by the target classifier, while keeping the L_p-distance from the initial example bounded:

$$\max_{x' \in \mathcal{F}(x, y)} \ell(f(x'), y) \quad \text{s.t.} \ \|x' - x\|_p \leq \varepsilon \quad (1)$$

This objective implicitly assumes that the adversary wants to keep the adversarial example as similar to the initial example as possible in terms of the examples' feature values. The closeness in terms of L_p distance aims to capture imperceptibility and to preserve the original example's semantics [21].

From Distances to Costs. To address the fact that imperceptibility or semantic similarity is not necessarily relevant for adversarial settings in tabular domains, we adapt the definition in Eq. (1) to the tabular setting by introducing a cost constraint.

This constraint represents the limited amount of resources available to the adversary to evade the target classifier. If the adversary can find an adversarial example that achieves this goal within the cost budget, the adversary proceeds with the attack. Formally, we associate a cost to the modifications needed to generate any adversarial example $x' \in \mathcal{F}(x, y)$ from the original example (x, y). We encode this cost as a function $c : \mathbb{X} \times \mathbb{X} \to \mathbb{R}^+$. We assume the generation cost is zero if and only if no change is enacted: $c(x, x) = 0$.

This formulation is generic: it can encompass geometric and semantic distances, but it goes beyond that. It exhibits the following desirable properties:

a. *Support for arbitrary feature types and rich semantics.* Whereas L_p distances only support numeric features, our generic cost model can support any feature type. This is because it does not enforce any structural constraints on the exact form of cost of changing a feature value x_i into x_i'. For example, the cost does not need to obey $|x_i - x_i'|$ as would be the case with L_1 distance. Moreover, unlike mathematical distances, our model does not require the costs to be symmetric. For instance, an increase in a feature value could have a different cost than a decrease.

b. *Enables more generic quantification of adversarial effort.* Our cost model imposes neither a geometric structure such as is the case with L_p distances, nor any ties to semantic similarity. Thus, the costs can be quantified in those units that are directly relevant to adversarial constraints. An important use case is that our model supports defining costs in the financial sense, i.e., assigning a dollar cost to mounting an attack with a given adversarial example as opposed to semantic closeness or closeness in feature space.

c. *Support for feature-level accumulation.* Related literature on attacks in tabular data often formalizes costs using indepenent per-feature constraints (see Section VII). Although our generic cost model supports such a special case, it also enables accumulation of per-feature costs. Therefore, it can encode a realistic assumption that changing more features increases adversary's expenditure.

The Optimization Problem. We assume that the cost-bounded adversary has a budget ε. The adversary aims to find any example that flips the classifier's decision *and* that is within the cost budget:

$$\max_{x' \in \mathcal{F}(x)} \mathbf{1}[F(x') \neq y] \quad \text{s.t.} \ c(x, x') \leq \varepsilon \quad (2)$$

Alternatively, the adversary can optimize a standard surrogate objective which ensures that the optimization problem can be

solved in practice:

$$\max_{x \in \mathcal{F}(x,y)} \ell(f(x), y) \quad \text{s.t. } c(x, x') \le \varepsilon, \qquad (3)$$

In the surrogate form, the optimization problem of the cost-bounded adversary is an adaptation of Eq. (1) with the norm constraint substituted by the adversarial-cost constraint. This formalization is in line with recent formalizations of adversarial examples [13], as opposed to early approaches which aim to find minimal-cost attacks [30]. This enables us to reuse tools from the recent literature on adversarial robustness to design defenses (see Section V).

B. Utility-Bounded Objective

The cost-bounded adversarial objective solves the issue of imperceptibility not being a suitable constraint for tabular data. It does not, however, tackle the problem of heterogeneity of examples: the adversary cannot assign different importance to different adversarial examples. In a realistic environment, it can be a serious drawback: an adversary might spend more resources than they get from a successful attack, or the defender suffers serious losses due to high-impact adversarial examples, even if for the majority of examples the defense is appropriate.

We propose to capture this heterogeneity by introducing the *gain* of an attack. The gain, $g(x^*) : \mathbb{X} \to \mathbb{R}^+$, represents the reward (e.g., the revenue) that the adversary receives if their attack using adversarial example x^* is successful.

We also introduce the concept of *utility*: the net benefit of deploying a successful attack. We define the utility $u_{x,y}(x^*)$ of an attack mounted with adversarial example x^* as simply the gain minus the costs:

$$u_{x,y}(x^*) \triangleq g(x^*) - c(x, x^*), \qquad (4)$$

where (x, y) is the initial example.

Recall that the adversary has black-box access to the target classifier. Thus, they can learn whether an example x^* evades the classifier or not (i.e., whether $F(x^*) \ne y$). Then, they can decide to deploy an attack with an adversarial example x^* only if the utility of the attack exceeds a given *margin* $\tau \ge 0$. Otherwise, the adversary discards this adversarial example. Formally, we can model this process by using a *utility constraint* instead of a cost constraint:

$$\max_{x \in \mathcal{F}(x,y)} \mathbf{1}[F(x) \ne y] \quad \text{s.t. } u_{x,y}(x) \ge \tau \qquad (5)$$

If we assume that the gain is constant for any adversarial example $x' \in \mathcal{F}(x,y)$ that is a modification of an initial example (x, y): $g(x) = g(x')$, this problem can also be seen as a variant of the cost-bounded problem in Eq. (2), where ε varies for different initial examples:

$$\max_{x' \in \mathcal{F}(x,y)} \mathbf{1}[F(x') \ne y]$$
$$\text{s.t. } u_{x,y}(x') \ge \tau$$
$$\iff g(x) - c(x, x') \ge \tau \qquad (6)$$
$$\iff c(x, x') \le \varepsilon(x) \triangleq g(x) - \tau$$

In Appendix A, we discuss the formalization of a utility-maximization objective which models an adversary which wants to maximize their profit subject to budget constraints.

C. Quantifying Cost and Utility

A natural question in our setup is how to define the adversary's costs and gains. This question is relevant to all related prior work on adversarial robustness in tabular data (see Section VII). For example, if adversarial robustness is defined in terms of an L_p distance, both the attacker and defender need to determine an acceptable perturbation magnitude, which inherently comes from domain knowledge.

In our applications (see Section VI), we focus on the settings in which adversarial capabilities are constrained in terms of financial costs. In such settings, we expect that the adversary is able to quantify the financial costs $c(x, x')$ and gains $g(x)$ by practical necessity. On the defender's side, estimating these values is trickier, as the defender might be unaware of the exact capabilities of the adversary. The defender thus needs to employ standard threat modeling techniques and domain knowledge. It is worth mentioning that the defender is not required to estimate the capabilities perfectly. Rather, they need to obtain the lower bound on the adversary's costs. After that, if the defended system is robust, it is robust against the adversary whose costs are at least as high as estimated.

In our utilitarian approach, it is possible to include other concerns and constraints of the adversary as part of the utility definition by measuring them in the same units as the utility (e.g., financial costs). For instance, as the driving concern behind imperceptibility-based approaches to adversarial robustness is the detection of an attack, the gain could be adjusted for a potential risk of being detected. The adversary could estimate the probability of being detected (e.g. using public statistics), and incorporate it into the gain by subtracting an expected value of the attack failure due to detection.

IV. FINDING ADVERSARIAL EXAMPLES IN TABULAR DOMAINS

In this section, we propose practical algorithms for finding adversarial examples suitable to achieve the adversarial objectives we introduce in Section III.

A. Graphical Framework

The optimization problems in Section III can seem daunting due to the large cardinality of $\mathcal{F}(x, y)$ when the feature space is large. To make the problems tractable, we transform them into graph-search problems, following the approach by Kulynych et al. [35]. Consider a *state-space graph* $\mathcal{G}(x) = (V, E)$. Each node corresponds to a feasible example in the feature space, $V = \mathcal{F}(x, y) \cup \{x\}$. Edges between two nodes x and x' exist if and only if they differ in value of one feature: there exists $i \in [n]$ such that $x_i \ne x'_i$, and $x_j = x'_j$ for all $j \ne i$. In other words, the immediate descendants of a node in the graph consist of all feasible feature vectors that differ from the parent in exactly one feature value.

Using this state-space graph abstraction, the objectives in Section III can be modeled as graph-search problems. Even though the graph size is exponential in the number of feature values, the search can be efficient. This is because it can construct the relevant parts of the graph on the fly as opposed to constructing the full graph in advance.

Building the state-space graph is straightforward when features take discrete values. To encode continuous features in the graph we discretize them by only considering changes to a continuous feature i that lie within a finite subset of its domain $\mathbb{X}_i$, in particular, on a discrete grid. The search efficiency depends on the size of the grid. As the grid gets coarser, finding adversarial examples becomes easier. This efficiency comes at the cost of potentially missing adversarial examples that are not represented on the grid but could fulfil the adversarial constraints with less cost or higher utility.

B. Attacks as Graph Search

In the remainder of the paper, we make the following assumptions about the adversarial model:

Assumption 1 (Modular costs). *The adversary's costs are modular: they decompose by features. Formally, changing the value of each feature i from x_i to x_i' has the associated cost $c_i(x_i, x_i') > 0$, and the total cost of modifying x into x' is a sum of individual feature-modification costs:*

$$c(x, x') = \sum_i^n c_i(x_i, x_i') \tag{7}$$

The state-space graph can encode modular costs by assigning weights to the graph edges. An edge between x and x' has an associated weight of $c_i(x_i, x_i')$, where i is the index of the feature that differs between x and x'. For pairs of examples $x^{(0)}$ and $x^{(t)}$ that differ in more than one feature, the cost $c(x^{(0)}, x^{(t)})$ is the sum of the edge costs along the shortest path from $x^{(0)}$ to $x^{(t)}$.

Assumption 2 (Constant gain). *For any initial example (x, y), the adversary cannot change the gain:*

$$\forall x' \in \mathcal{F}(x, y): \quad g(x) = g(x') \tag{8}$$

This follows the approach in utility-oriented strategic classification (as detailed in Section III-B). This assumption is not formally required for our attack algorithms (described next in this section), but we focus on this setting in our empirical evaluations. In the extended version of this paper*, we experimentally show that removing this assumption does not significantly affect our results.

Strategies to Find Adversarial Examples. Under our two assumptions, the cost-bounded objective in Eq. (2) and the utility-bounded objective in Eq. (6) can be achieved by finding any adversarial example that is classified as target class and is within a given cost bound. Thus, these adversarial goals can be achieved using *bounded-cost search* [36].

*arxiv.org/abs/2208.13058

Algorithm 1 Best-First Search (BFS)

1: **function** $\text{BFS}_{B,s,\varepsilon}(x)$
2:     open $\leftarrow$ MINPRIORITYQUEUE$_B(x, 0)$
3:     closed $\leftarrow \{\}$
4:     **while** open is not empty **do**
5:         $v \leftarrow$ open.POP()
6:         **if** $v \notin$ closed **then**
7:             CLOSED $\leftarrow$ CLOSED $\cup \{v\}$
8:         **if** $\eta(v) \geq \delta$ **then return** v
9:         $S \leftarrow$ EXPAND(v)
10:        **for** $t \in S$ **do**
11:           **if** $t \notin$ closed and $c(x, t) \leq \varepsilon$ **then**
12:             open.ADD$(t, s(v, t))$

We start with the *best-first search* (BFS) [35, 37], a flexible meta-algorithm that generalizes many common graph search algorithms. In its generic version (Algorithm 1) BFS keeps a bounded priority queue of *open nodes*. It iteratively pops the node v with the highest score value from the queue (best first), and adds its immediate descendants to the queue. This is repeated until the queue is empty. The algorithm returns the node with the highest score out of all popped nodes.

The BFS algorithm is parameterized by the *scoring function* $s: V \times V = \mathbb{X} \times \mathbb{X}$ and the size of the priority queue B. Different choices of the scoring function yield search algorithms suited for solving different graph-search problems, such as Potential Search for bounded-cost search [36, 38], and A* [39, 40] for finding the minimal-cost paths. When $B = \infty$, the algorithm might traverse the full graph and is capable of returning the optimal solution. As the size of B decreases, the optimality guarantees are lost. When $B = 1$ BFS becomes a *greedy* algorithm that myopically optimizes the scoring function. When $1 < B < \infty$ we get a *beam search* algorithm that keeps B best candidates at each iteration.

To achieve the adversarial objectives in Section III, we propose to use a concrete instantiation of BFS, what we call the *Universal Greedy (UG)* algorithm. Inspired by heuristics for cost-bounded optimization of submodular functions [41, 42], we set the scoring function to balance the increase in the classifier's score and the cost of the change:

$$s(v, t) = -\frac{f(t) - f(v)}{c(v, t)} \tag{9}$$

The minus sign appears because BFS expands the lowest scores first, and we need to maximize the score. We set the beam size to $B = 1$ (greedy), which enables us to find high-quality solutions to *both* cost-bounded and utility-bounded problems at reasonable computational costs (see Section VI).

V. Defending from Adversarial Examples in Tabular Domains

A common way to mitigate the risks of adversarial examples is adversarial training [9, 13]. In a standard approach [13], these adversarial examples are constructed during training by

5

modifying natural examples x with perturbations constrained in an L_p-ball of radius ε as in Eq. (1).

The distance function and ε encode the *threat model* that adversarial training aims to defend against. The choice of the distance function depends on the characteristics of the input domain. As explained in Sections I and III, this approach does not apply to the tabular domains. In tabular domains, imperceptibility is not necessarily a relevant constraint. Instead, the adversary's actions are constrained by feasibility and the cost of the transformations. Moreover, the tabular input domain is often a mix of discrete and continuous features, as opposed to continuous or quantized in, e.g., image domains, where adversarial examples are mostly studied.

Another difference between the image and tabular domains is the efficiency of generating adversarial examples. In images, adversarial examples used for training are generated using efficient methods such as Projected Gradient Descent (PGD) [13] or the Fast Gradient Sign Method (FGSM) [9, 43]. These algorithms produce adversarial examples fast, and enable the efficient implementation of adversarial training. Fast generation, however, is not possible for tabular domains. The algorithms to produce tabular adversarial examples introduced in Section IV require thousands of inference operations using the target model. Although generating one example can take seconds, generating thousands of adversarial examples required during adversarial training quickly becomes infeasible for computationally constrained defenders. To make the generation of adversarial examples feasible during adversarial training, we introduce approximate versions of the attacks that rely on a relaxation of initial attack constraints.

A. Relaxing the Constraints

Following the setting of the standard Projected Gradient Descent (PGD) method [13], adversarial training for the cost-bounded adversary could be defined as the following optimization objective:

$$\min_{\theta} \max_{x' \in \mathcal{F}(x,y)} \ell(f_{\theta}(x'), y) \quad \text{s.t. } c(x, x') \leq \varepsilon, \quad (10)$$

where f_{θ} is a parametric classifier and θ are its parameters.

To keep the computational requirements low, we relax the problem to optimize over a convex set, which enables us to adapt the PGD method. Let us define B_{ε} to be the constraint region of Eq. (10):

$$B_{\varepsilon}(x,y) \triangleq \{x' \in \mathcal{F}(x,y) \mid c(x, x') \leq \varepsilon\}$$

We construct a relaxation of B_{ε} in two steps:

$$B_{\varepsilon} \xrightarrow[(1)]{} \bar{B}_{\varepsilon} \xrightarrow[(2)]{} \tilde{B}_{\varepsilon}$$

(1) *Continuous relaxation.* We map B_{ε} into a continuous space using an *encoding function* $\phi : \mathbb{X}^n \to \mathbb{R}^m$, and a *relaxed cost function* $\bar{c} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^+$. The continuous relaxation is defined as:

$$\bar{B}_{\varepsilon} \triangleq \{\phi(x') \mid \bar{c}(\phi(x), \phi(x')) \leq \varepsilon\}, \quad (11)$$

where $x' \in \mathcal{F}(x,y)$. The pair $(\phi, \bar{c})$ is designed to satisfy the following condition:

$$\forall x' \in B_{\varepsilon}(x,y): \bar{c}(\phi(x), \phi(x')) \leq c(x, x'), \quad (12)$$

ensuring that every example $x' \in B_{\varepsilon}(x,y)$ is mapped to an element in the relaxed set, $\phi(x') \in \bar{B}_{\varepsilon}$. We denote the encoded value $\phi(x)$ as $\bar{x}$.

(2) *Convex cover.* To enable adversarial training using PGD, we need that elements of the relaxed set can be projected onto the constraint region. For this purpose, we cover $\bar{B}_{\varepsilon}$ with a convex superset $\tilde{B}_{\varepsilon}$, e.g., a convex hull of $\bar{B}_{\varepsilon}$. The convex superset $\tilde{B}_{\varepsilon}$ needs to be constructed such that there exists an efficient algorithm for *projection*: For a given (x, y), and a point $t \in \mathbb{R}^m$, we want to be able to efficiently solve $\min_{t' \in \tilde{B}_{\varepsilon}(x,y)} \|t - t'\|_2$.

Encoding and Cost Functions. As we assume that the cost of modifications is modular (see Section IV-B), we define the encoding ϕ and cost $\bar{c}$ functions to be modular too:

$$\phi(x) = [\phi_1(x_1), ..., \phi_n(x_n)]$$

$$\bar{c}(\phi(x), \phi(x')) = \sum_{i=1}^{n} \bar{c}_i(\phi_i(x_i), \phi_i(x_i'))$$

With this formulation, the problem of constructing suitable ϕ and $\bar{c}$ functions is reduced to finding $\phi_i : \mathbb{X}_i \to \mathbb{R}^m$ and $\bar{c}_i$ for each feature. If for all i both ϕ_i and $\bar{c}_i$ individually fulfill the requirement in Eq. (12):

$$\bar{c}_i(\phi_i(x_i), \phi_i(x_i')) \leq c_i(x_i, x_i'),$$

then the modular cost $\bar{c}(\bar{x}, \bar{x}')$ fulfills Eq. (12) as well.

In the following we introduce ϕ and $\bar{c}$ functions for categorical and numeric features.

Categorical features. As the encoding function $\phi(x_i)$ for categorical features we use standard one-hot encoding. As the cost function for categorical features, we define $\bar{c}_i$:

$$\bar{c}_i(\bar{x}_i, \bar{x}_i') = \min_{t \in \mathcal{F}(x,y)} c_i(x_i, t) \cdot \frac{1}{2} \|\bar{x}_i - \bar{x}_i'\|_1,$$

where $\mathcal{F}_i(x, y)$ is the set of feasible values of the feature i. For example, let x_i be a categorical feature with 4 possible values $X_i = \{a, b, c, d\}$, and let the minimal cost of change be 2. When $x_i = b$ and $x' = c$ ($\bar{x}_i = (0, 1, 0, 0)$, $\bar{x}_i' = (0, 0, 1, 0)$ after one-hot encoding). Then, $\bar{c}_i(\bar{x}_i, \bar{x}_i') = \frac{1}{2} \cdot 2 \cdot 2 = 2$.

This cost function enables us to perform the two-step relaxation described before. First, it satisfies Eq. (12), and therefore the constraint region $\bar{B}_{\varepsilon}$ includes all mapped examples of B_{ε}. Second, we can obtain the convex superset $\tilde{B}_{\varepsilon}$ as a continuous L_1 ball around the mapped values $\bar{x} \in \bar{B}_{\varepsilon}$.

Numeric features. A numeric feature is a feature with values belonging to an ordered subset of $\mathbb{R}$ (e.g. integer, real). In most cases, the identity function ($\phi(x_i) = x_i$) is sufficient for numerical features. However, more complex encoding functions could also be desirable. For example, when one needs to reduce numerical errors, which can be achieved by normalizing the feature values to $[-1, 1]$, or when the cost is non-linear.

In general, projecting onto arbitrary sets can be challenging. Specifically, the constraint region B_ε could be non-convex, e.g., hypothetically, if the cost is a pathological function such as the Dirichlet function. We therefore must limit the scope of possible adversarial cost functions that we can model during adversarial training to those that are compatible with efficient projection. For this, we introduce a cost model that covers a broad class of functions for which $c_i(x_i, x_i')$ can be expressed as $K_i \cdot |\psi(x_i) - \psi(x_i')|$, where K_i is a constant and $\psi(x)$ is an invertible function.

For instance, this model covers the following exponential cost model: $c(x, x') = K \cdot |e^x - e^{x'}|$. In this case, we can encode the features as $\phi(x_i) = \psi^{-1}(x_i) \triangleq \ln(x_i)$. This transformation enables us to account for certain non-linear cost functions c with respect to the input space using linear cost functions $\bar{c}$ in the relaxed space $\bar{B}_\varepsilon$.

We define the cost function for numerical features as a piecewise-linear function, with different coefficients for increasing or decreasing the feature value:

$$\bar{c}_j(\bar{x}_j, \bar{x}_j') = c_{j-}(x) \cdot [\bar{x}_j - \bar{x}_j']^+ + c_{j+}(x) \cdot [\bar{x}_j' - \bar{x}_j]^+$$

where $[t]^+$ returns t if $t > 0$, and 0 otherwise, and $c_{j-}(x)$ and $c_{j+}(x)$ encode the costs for decreasing and increasing the value of the feature j, respectively, and can vary from one initial example x to another.

Note that in this model the final cost of a modification could depend on the way in which this modification is achieved. A direct modification from x to x'' could have different cost than first modifying x to x' and then x' to x'', i.e., $\bar{c}_i(\bar{x}, \bar{x}'') \neq \bar{c}_i(\bar{x}, \bar{x}') + \bar{c}_i(\bar{x}', \bar{x}'')$.

Total cost. Given the set of categorical feature indices, $\mathcal{C}$, and the set of numeric feature indices, $\mathcal{I}$, the total cost function is:

$$\bar{c}(\bar{x}, \bar{x}') = \sum_{i \in \mathcal{C}} \min_{t \in \mathcal{F}_i(x, y)} c_i(x_i, t) \cdot \frac{1}{2} \|\bar{x}_i - \bar{x}_i'\|_1 \\ + \sum_{j \in \mathcal{I}} c_{j-}(x) \cdot [\bar{x}_j - \bar{x}_j']^+ + c_{j+}(x) \cdot [\bar{x}_j' - \bar{x}_j]^+ \quad (13)$$

B. Adversarial Training with Projected Gradient Descent

Using the cost model introduced before, we redefine the training optimization problem in Eq. (10) to generate adversarial examples over a specific instantiation of the convex set $\tilde{B}_\varepsilon$, as follows:

$$\min_\theta \max_{\tilde{x}' \in \tilde{B}_\varepsilon(x, y)} \ell(f_\theta(\tilde{x}'), y), \quad (14)$$

where we specify $\tilde{B}_\varepsilon$ as:

$$\tilde{B}_\varepsilon(x, y) \triangleq \{\bar{x} + \delta \mid \delta \in \mathbb{R}^m \wedge \bar{c}(\bar{x}, \bar{x} + \delta) \le \varepsilon\}. \quad (15)$$

Thus, we can rewrite Eq. (14):

$$\min_\theta \max_{\delta \in \mathbb{R}^m} \ell(f_\theta(\bar{x} + \delta), y) \\ \text{s.t. } \bar{c}(\bar{x}, \bar{x} + \delta) \le \varepsilon \quad (16)$$

This objective can be optimized using standard PGD-based adversarial training [13]. Due to the construction of our cost function in Eq. (13), we can use existing algorithms for projecting onto a weighted L_1-ball [44, 45] with an appropriate choice of weights. As these approaches are standard, we omit them in the main body, and provide the details in Appendix B.

C. Adversarial Training against a Utility-Bounded Adversary

For the utility-bounded adversary we propose to use an objective similar to Eq. (16), but applying individual constraints to different examples:

$$\min_\theta \max_{\delta \in \mathbb{R}^m} \ell(f_\theta(\bar{x} + \delta), y) \\ \text{s.t. } \bar{c}(\bar{x}, \bar{x} + \delta) \le \varepsilon(x) \triangleq [\bar{g}(\bar{x} + \delta)]_+, \quad (17)$$

where $\bar{g}(\bar{x})$ is such that $\bar{g}(\bar{x}) = g(x)$. In this formulation, we apply our assumption of invariant gain (see Section IV-B) as $\bar{g}(\bar{x} + \delta) = \bar{g}(\bar{x})$.

This objective aims to decrease the adversary's utility by focusing the protection on examples with high gain. The main difference with respect to the cost-constrained objective in Eq. (16) is that here we use a different cost bound for different examples $\varepsilon(x)$. This formulation enables us to directly use the PGD-based adversarial training to defend against utility-bounded adversaries as well.

VI. EXPERIMENTAL EVALUATION

In this section, we show that our graph-based attacks can be used by adversaries to obtain profit, and that our proposed defenses are effective at mitigating damage from these attacks.

A. Experimental Setup

1) Datasets: We perform our experiments on three tabular datasets which represent real-world applications for which adversarial examples can have social or economic implications:

- `TwitterBot` [46]. The dataset contains information about more than 3,400 Twitter accounts either belonging to humans or bots. The task is to detect bot accounts. We assume that the adversary is able to purchase bot accounts and interactions on darknet markets, thus modifying the features that correspond to the account age, number of likes, and retweets.
- `IEEECIS` [47]. The dataset contains information about around 600K financial transactions. The task is to predict whether a transaction is fraudulent or benign. We model an adversary that can modify three features for which we can outline the hypothetical method of possible modification, and estimate its cost: payment-card type, email domain, and payment-device type.
- `HomeCredit` [48]. The dataset contains financial information about 300K home-loan applicants. The main task is to predict whether an applicant will repay the loan or default. We use 33 features, selected based on the best solutions to the original Kaggle competition [48]. Of these, we assume that 28 can be modified by the adversary, e.g., the loan appointment time.

2) Models: We evaluate our attacks against three types of ML models commonly applied to tabular data. First, an L_2-regularized *logistic regression (LR)* with a regularization parameter chosen using 5-fold cross-validation. Second, *gradient-boosted decision trees (XGBT)*. Third, *TabNet* [49], an attentive transformer neural network specifically designed for tabular data. We optimize the number of steps as well as the capacity of TabNet's fully connected layers using grid search.

3) Adversarial Features: We assume that the feasible set consists of all positive values of numerical features and all possible values of categorical features. For simplicity, we avoid features with mutual dependencies and treat the adversarially modifiable features as independent. We detail the choice of the modifiable features and their costs in Section C-B.

4) Metrics: To evaluate the effectiveness of the attacks and defenses, we use three main metrics:

- *Adversary's success rate:* The proportion of correctly classified examples from a test set X_{test} for which adversarial examples successfully generated using the attack algorithm $\mathcal{A}(x, y)$ evade the classifier:

$$\Pr_{(x,y) \sim X_{\text{test}}} [F(\mathcal{A}(x,y)) \neq y \wedge F(x) = y].$$

- *Adversarial cost:* Average cost of successful adversarial examples:

$$\mathbb{E}_{(x,y) \sim X_{\text{test}}} [c(x, \mathcal{A}(x,y)) \mid F(\mathcal{A}(x,y)) \neq y \wedge F(x) = y].$$

- *Adversarial utility:* Average utility (see Eq. (4)) of successful adversarial examples:

$$\mathbb{E}_{(x,y) \sim X_{\text{test}}} [u_{x,y}(\mathcal{A}(x,y)) \mid F(\mathcal{A}(x,y)) \neq y \wedge F(x) = y].$$

In all cases, we only consider correctly classified initial examples which enables us to distinguish these security metrics from the target model's accuracy. We introduce additional metrics in the experiments when needed.

B. Attacks Evaluation

We evaluate the attack strategy proposed in Section IV in terms of its effectiveness, and empirically justify its design.

1) Design Choices of the Universal Greedy Algorithm: When designing attack algorithms in the BFS framework (see Algorithm 1) there are two main design choices: the scoring function, and the beam size. We explore different configurations and show that our parameter choices for the Universal Greedy attack produce high-quality adversarial examples.

Beam size. We define the beam size of the Universal Greedy attack to be one. The other options that we evaluate are 10 and 100. We evaluate by running three types of attacks: cost-bounded for three cost bounds ε, and utility-bounded at the breakeven margin $\tau = 0$. We compute two metrics: attack success, and the success-to-runtime ratio. This ratio represents how much time is needed to achieve the same level of success rate using each choice of the beam size. This metric is more informative for our evaluation than runtime, as the runtime is just proportional to the beam size.

For feasibility reasons, we use two datasets: `TwitterBot` and `IEEECIS`. We aggregate the metrics across the three models (LR, XGBT, TabNet), and report the average. The results on `TwitterBot` are equivalent to the results on `IEEECIS`, thus for conciseness we only report `IEEECIS` results.

We find that the success rates are equal up to the percentage point for all choices of the beam size. We show the detailed numeric results in Table VII in the Appendix. As the smallest beam size of one is the fastest to run, it demonstrates the best success/time ratio, therefore, is the best choice.

Scoring function. Recall from Eq. (9) that the scoring function is the cost-weighted increase in the target classifier's confidence. which aims to maximize the increase in classifier confidence at the lowest cost.

Other choices for the scoring function $s(v, t)$ could be:

- A^* *algorithm* [35, 39, 40]: $s(v, t) = c(v, t) + \lambda \cdot h(t)$, where $h(t)$ is a heuristic function, which estimates the remaining cost to a solution and $\lambda > 0$ is a greediness parameter [50]. This scoring function balances the current known cost of a candidate and the estimated remaining cost. We choose the model's confidence for the positive class, $h(x) = f(x)$, as a heuristic function. Intuitively, this works as a heuristic, because the lower the confidence for the positive class, the more likely we are close to a solution: an example classified as the target class.
- *Potential Search* (PS) [36, 38]: $s(v, t) = {}^{h(t)}/_{(\varepsilon - c(v,t))}$, which additionally takes into account the cost bound ε, thus becoming more greedy (i.e., optimizing $s(v, t) = \lambda \cdot h(t)$ with $\lambda \approx {}^1/_\varepsilon$) when the cost of the current candidate leaves a lot of room within the ε budget. We also choose $h(x) = f(x)$ as a heuristic function.
- *Basic Greedy*: $s(v, t) = {}^{-f(t)}/_{c(s,t)}$, which aims to maximize the classifier's confidence, yet balance it with the incurred cost. Unlike Eq. (9), this scoring function does not care about the relative increase of the confidence, only about its absolute value.

We evaluate the choice of the scoring function on the `TwitterBot` and `IEEECIS` datasets, with the beam size fixed to one. We run the cost-bounded and utility-bounded attacks in the same configuration as before, and measure two metrics averaged over the models: Attack success, and attack success/time ratio.

Table I shows the results. On `IEEECIS`, the Universal Greedy outperforms the other choices in terms of success rate and the success/time ratio. On the `TwitterBot` dataset, it outperforms the other choices in the utility-bounded and unbounded attacks. For cost-bounded attacks, the Universal Greedy offers very close performance to the best option, the Basic Greedy.

2) Graph-Based Attacks vs. Baselines: We compare the Universal Greedy (UG) algorithm against two baselines: previous work, and the minimal-cost adversarial examples.

Previous Work: PGD. As our cost model differs from the existing approaches to attacks on tabular data, we fundamentally cannot perform a fully apples-to-apples comparison against existing attacks (see Section IV). To compare against the high-level ideas from prior work, we follow the spirit of the attack

	Adv. success, %			
Cost bound →	10	30	Gain	∞
Scoring func. ↓				
UG	**45.32**	**56.57**	**56.22**	**68.20**
A*	42.37	55.62	55.34	53.47
PS	**45.32**	55.14	56.18	N/A
Basic Greedy	42.37	55.46	55.38	53.82

	Success/time ratio			
Cost bound →	10	30	Gain	∞
Scoring func. ↓				
UG	**3.78**	**4.80**	**2.53**	**2.06**
A*	3.29	3.83	1.89	1.15
PS	**3.78**	4.01	2.26	N/A
Basic Greedy	3.21	3.86	2.01	1.16

(a) IEEECIS

	Adv. success, %			
Cost bound →	1,000	10,000	Gain	∞
Scoring func. ↓				
UG	80.24	**85.35**	**21.63**	**87.00**
A*	77.56	84.45	20.29	86.25
PS	79.95	85.19	21.48	N/A
Basic Greedy	**80.40**	85.04	**21.63**	86.85

	Success/time ratio			
Cost bound →	1,000	10,000	Gain	∞
Scoring func. ↓				
UG	208.95	205.76	**64.99**	**205.31**
A*	206.33	201.93	62.25	201.31
PS	205.85	203.18	63.76	N/A
Basic Greedy	**210.20**	**206.20**	64.32	204.96

(b) Twitter bot

TABLE I: *Effect of the scoring-function choice* for graph-based attacks on IEEECIS. In all settings, our Universal Greedy scoring function offers the best success rate and performance.
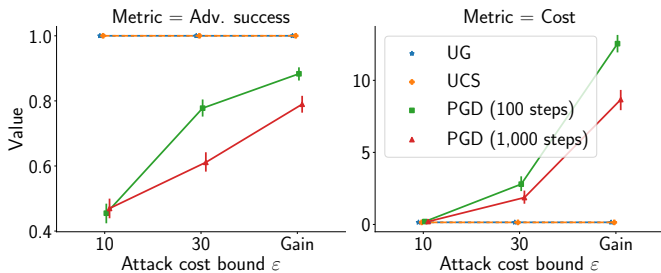

Fig. 1: *Universal Greedy attack vs Baselines.* Left: Attack success rate (higher is better for the adversary). Right: Attack cost (lower is better for the adversary). For all cost bounds, our graph-based attack outperforms standard PGD and returns close to optimal-cost adversarial examples (obtained with Uniform-Cost Search, UCS).

by Ballet et al. [51], which modifies the standard optimization problem from Eq. (1) to use correlation-based weights. We adapt the standard L_1-based PGD attack [13, 52] to (1) support categorical features through discretization, and (2) use weighted L_1 norm following our derivations in Section V-A. We provide a detailed description of this adaptation in Algorithm 3 in Appendix C.

We run attacks using PGD with 100 and 1,000 steps, and compare it to UG (Section IV) on the TwitterBot and IEEECIS datasets. As PGD can only operate on differentiable models, in this comparison we only evaluate the performance of the attacks against TabNet.

We run the cost-bounded attacks using two bounds ε, that are specific to each dataset (see Appendix C for the exact attack parameters). As before, we also run a utility-bounded attack at the breakeven margin $\tau = 0$. We measure the success rates of the attacks, as well as the average cost of the obtained adversarial examples. For conciseness, we do not report the results on TwitterBot, as they find they are equivalent to those on IEEECIS.

Fig. 1 shows that the UG attack consistently outperforms the PGD-based baseline both in terms of the success rate and the costs. Our attacks are superior even when the PGD-based baseline produces feasible adversarial examples.

Minimal-Cost Adversarial Examples. As UG is a greedy algorithm, we additionally evaluate how far are the obtained adversarial examples from the optimal ones in terms of cost. For this, we compare the results from UG to a standard Uniform-Cost Search (UCS) [35]. UCS is an instantiation of the BFS framework (see Section IV) with unbounded beam size, and the scoring function equal to the cost: $s(v, t) = c(v, t)$. In our setting, UCS is guaranteed to return optimal solutions to the following optimization problem:

$$\min_{x' \in \mathcal{F}(x,y)} c(x, x') \quad \text{s.t.} \ F(x') \neq y \quad (18)$$

Fig. 1 shows that UG has almost no overhead over the minimal-cost adversarial examples on TabNet ($1.03\times$ overhead on average). In fact, the average and median cost overhead is $1.80\times$ and $1\times$ over all models, respectively. There exist some outlier examples, however, with over $100\times$ cost overhead. We provide more information on the distribution of cost overhead in Appendix C.

3) Performance against Undefended Models: Having shown that the attacks outperform the baseline, and the design choices are sound, we demonstrate that the attacks bring some *utility* to the adversary. In this section, we evaluate the attacks in a non-strategic setting: the models are not deliberately defended against the attacks. For conciseness, we only evaluate cost-bounded attacks, as the next section provides an extensive demonstration of utility-bounded attacks.

In *all* evaluated settings, the attacks have non-zero success rates and achieve non-zero adversarial utility. Fig. 2 show the results of cost-bounded attacks for IEEECIS and HomeCredit datasets. We omit the results for LR on HomeCredit as this model does not perform better than the random baseline. An average adversarial example obtained using the cost-bounded objective brings a profit of \$125 to the adversary when attacking

(a) `IEEECIS`. Model (test acc.): ● LR (0.62) ● XGBT (0.83) ● TabNet (0.77)

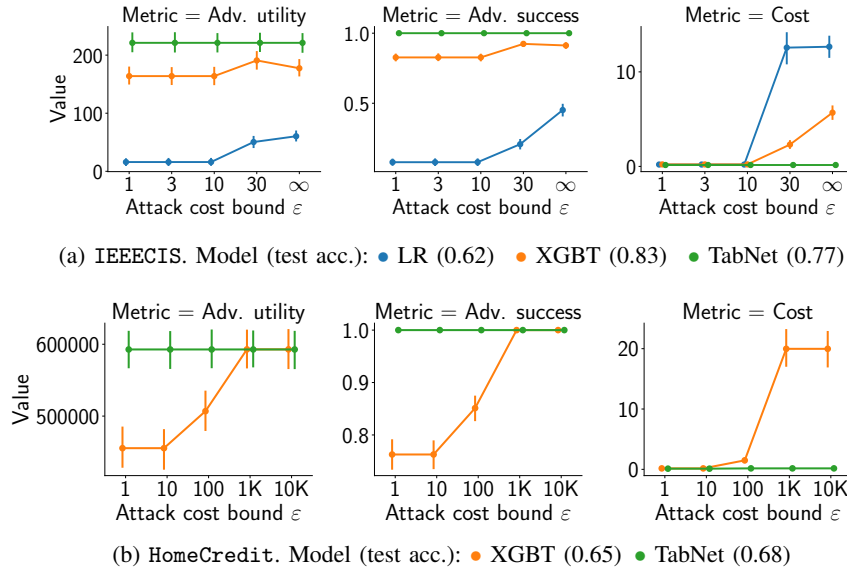(b) `HomeCredit`. Model (test acc.): ● XGBT (0.65) ● TabNet (0.68)

Fig. 2: Results of cost-bounded graph-based attacks against three types of models. Left pane: Adversarial utility (higher is better for the adversary). Middle and right panes: See Fig. 1. On `IEEECIS`, the attack can achieve utility from approximately \$10 to \$125 per attack depending on the target model. On `HomeCredit`, the average utility ranges between \$400,000 and \$600,000.

the `IEEECIS` TabNet model, and close to 100% of examples in the test data can be turned into successful adversarial examples.

Although for all models we see non-zero success and utility, some models are less vulnerable than others, even without any protection. For example, the success rate of the adversary against LR on `IEEECIS` is much lower than against TabNet (at least 50 p.p. lower). This model, however, is also comparatively inaccurate, with only 62% classification accuracy.

C. Evaluation of Our Defense Methods

We evaluate the defense mechanisms proposed in Section V in two scenarios. First, a scenario in which the adversary's objective used by the defender for adversarial training—cost-bounded (CB) or utility-bounded (UB)—matches the attack that will be deployed by the adversary. Second, a scenario in which the defender models the adversary's objective incorrectly, and uses a different attack than the adversary when performing adversarial training.

Baselines. We set two comparison baselines which provide boundaries for which a defense can be considered effective. On the accuracy side, we consider the *clean baseline*: a model trained without any defense. It provides the best accuracy, but also the least robustness. Any defense that does not achieve at least the clean baseline's *robustness* should not be considered, as the clean baseline would always provide better or equal accuracy, thus a better robustness-accuracy trade-off. On the robustness side, we consider the *robust baseline*: a model for which all features that can be changed by the adversary are masked with zeroes for training and testing. As this removes any adversarial input, this model is invulnerable to attacks within the assumed adversarial models. Any practical defense

must outperform the robust baseline in terms of *accuracy*. Otherwise, the robust baseline would provide a better trade-off.

Table II shows the clean and robust baselines' accuracy for the three datasets. On `TwitterBot`, the robust baseline performs almost as well as the clean model. As there is no space for a better defense for `TwitterBot`, we only evaluate our defenses for the `IEEECIS` and `HomeCredit` models.

We train our attacks and defenses using the parameters listed in Table III in Appendix C.

1) Defender matches the adversary: We first evaluate the case in which the adversarial training used to generate the defense is perfectly tailored to the adversary's objective.

Cost-Bounded Defense vs. Cost-Bounded Attack. Fig. 3 shows the results when defender and adversary use CB objectives. For both `IEEECIS` and `HomeCredit` the CB-trained defense is effective when the adversary uses CB attacks: the adversary only finds successful adversarial examples with positive utility if they invest more than the budget assumed by the defender. If the defender greatly underestimates the adversary's budget of the adversary (e.g., training with $\varepsilon = 10$ when the adversary's budget is $\varepsilon = 1000$), the adversary obtains a high profit. Therefore, an effective defense requires an adequate estimation of the adversary's capabilities.

Utility-Bounded Defense vs. Utility-Bounded Attack. Fig. 4 shows the results of our evaluation when the defender and the adversary use UB objectives. The defense is effective: it
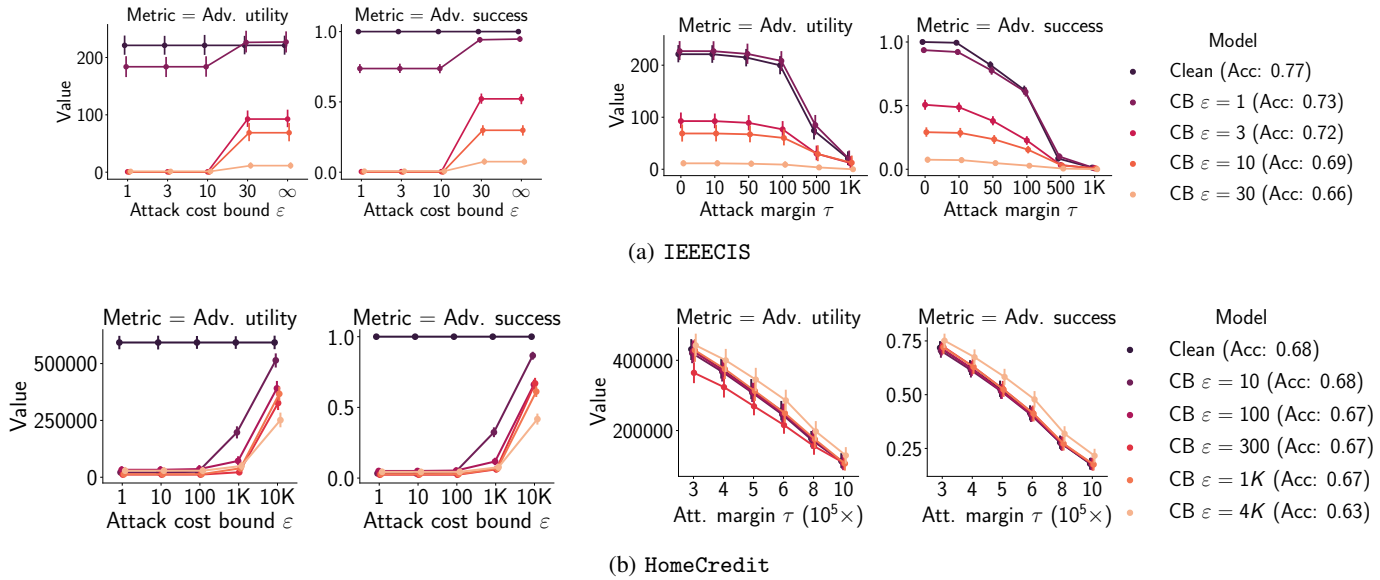
TABLE II: Baseline performance

Accuracy	`TwitterBot`	`IEEECIS`	`HomeCredit`
Clean baseline	0.775	0.755	0.680
Robust baseline	0.773	0.685	0.556
Random baseline	0.566	0.500	0.501

(a) `IEEECIS`

(b) `HomeCredit`

Fig. 3: *Cost-bounded adversarial training* for different adversarial budgets ε. Evaluation against cost-bounded (left), and utility-bounded (right) attacks. We represent the adversary's success and utility (y-axis) versus the adversary's attack budget ε or desired utility margin τ (x-axis). CB attacks only have substantial success and profit when the adversary invests more than the budget assumed by the defender. UB attacks are thwarted for `IEEECIS`, but CB training is not significantly effective on `HomeCredit`, and for some models even enables higher adversary's utility.

decreases both the success rate and the adversary's utility on both datasets. On `IEEECIS`, the adversary can only succeed when their desired profit τ is smaller than the τ used to train the defense. On `HomeCredit`, we observe a similar behaviour, although when training for margins τ less than 500K, the model does not completely mitigate adversaries that wish to have larger profits. When the defender allows for large adversary's profit margins (e.g., $\tau = 800K$ or $\tau = 1M$), the models become significantly robust with little accuracy loss.

2) Defender Does not Match the Adversary: In the previous section, we show that if the defender correctly models the attacker's objective, our defenses offer good robustness. Next, we evaluate the performance when the defender's model does not match the adversary's objective. This is likely to happen in realistic deployments, as the defender might not have any a priori knowledge of the adversary's objective.

Utility-Bounded Defense vs. Cost-Bounded Attack. Fig. 4 shows our evaluation results when a CB adversary attacks a defense trained assuming UB objectives. For both datasets, the robustness improves with respect to the clean baseline, even though robustness against CB adversaries is not the defense goal. The improvement is more pronounced as the defender tightens the profit margin (decrease in τ). This effect is stronger on `HomeCredit`, where even loose profit margins provide significant robustness. The adversary can increase their success (on both datasets) and utility (only `HomeCredit`) by increasing their budget ϵ. These experiments show that UB training improves robustness *even when the adversary has a different objective.*

Cost-Bounded Defense vs. Utility-Bounded Attack. When a CB defense confronts a UB adversary, we observe a different behaviour, showed in Fig. 3. On `IEEECIS`, CB adversarial training increases the robustness of the model against UB adversaries, with greater effect as the cost bound increases. When protecting against high adversary's budgets ($\varepsilon = 30$), however, the impact on accuracy is too large, and the robust baseline becomes preferable. For `HomeCredit`, the situation is worse. Although performance is always above the robust baseline, we observe little improvement with respect to the clean model. Even worse, for certain parameters, the utility of the adversary can increase after adversarial training (see the model trained with a bound of $\varepsilon = 4000$). We conclude that CB training is not effective against an adversary with a different objective.

3) Robustness-Accuracy Trade-offs: In the previous sections, we evaluate the effectiveness of the defenses depending on the adversary's and defender's objectives. We now evaluate the trade-offs between defense effectiveness in reducing the adversary's performance on one hand, and the accuracy of the model on the other hand.

As adversarial training penalizes model's sensitivity to modifications of input features, it results in certain features having less influence on the output. These features cannot be used for prediction to the same extent as features in the clean baseline, which leads to the degradation of the model's accuracy. On the positive side, these features can neither be used by the adversary—the robust baseline being extreme in which all features prone to manipulation are zeroed—reducing the attack's success and utility.
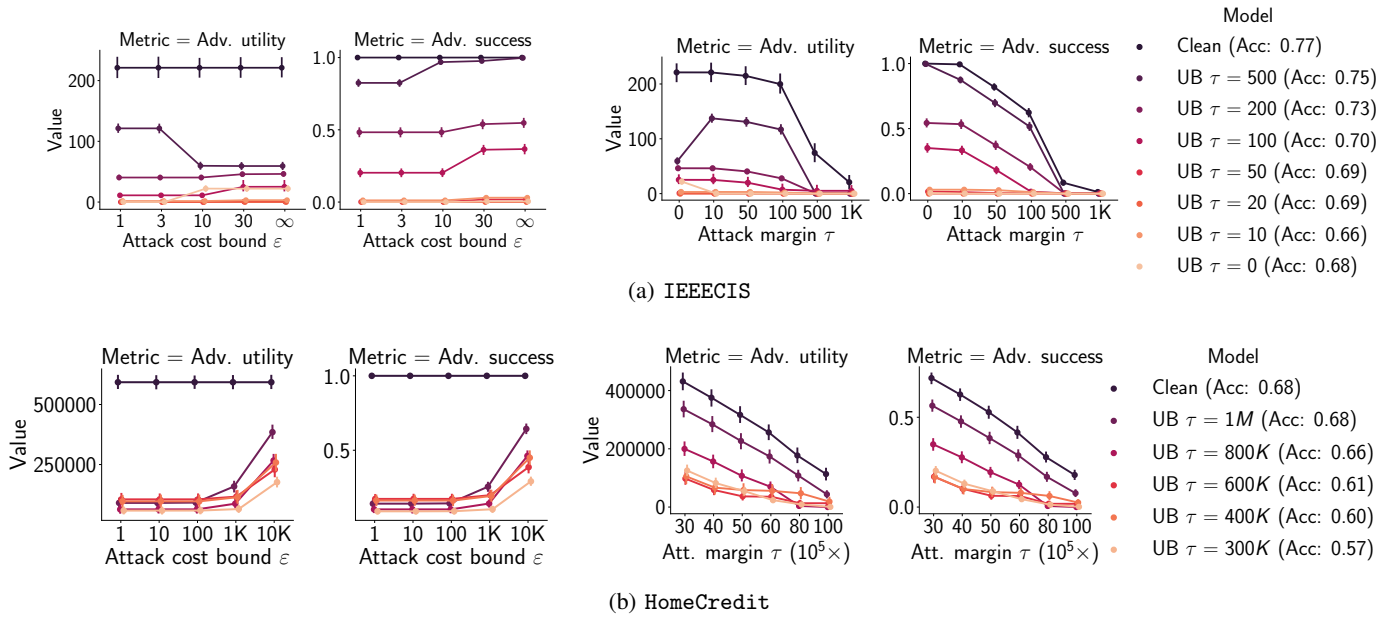
(a) `IEEECIS`

(b) `HomeCredit`

Fig. 4: *Utility-Bounded adversarial training* for different adversarial utility margins τ. Evaluation against cost-bounded (left) and utility-bounded (right) adversaries. We show the adversary's success and utility (y-axis) versus the adversary's attack budget ϵ or desired margin τ (x-axis). On `HomeCredit`, the UB training decreases the performance of both UB and CB attacks, being robustness better against the former. Even when enabling a large profit margin ($\tau = 1M$) the attack success rate decreases by 40%, at the same time not affecting the accuracy.

In our experiments, we observe different nature of these trade-offs for robustness against CB and UB adversaries. Against the CB adversary, the accuracy-robustness trade-off depends on the modeled adversary's budget, and we do not observe that either defense approach is superior to another. On the contrary, against the UB adversaries, we consistently observe better robustness (less adversarial utility for the same accuracy) for the UB defense compared to the CB one. We present the detailed accuracy-robustness plots in Fig. 6 in Appendix C.

Thus, in the absence of knowledge of the adversary's objective, utility-bounded defenses are preferable. They outperform CB adversarial training when the adversary is utility-oriented, and offer comparable performance against CB attacks.

VII. RELATED WORK

Our conceptual contributions span three aspects of adversarial robustness in tabular domains: new formulations of (A) *adversarial objectives*, (B) *attack strategies* within these objectives, and (C) *adversarial training*-based defenses. We review the related works in each of the aspect next. We also provide a concise summary in the Appendix (Table VIII).

A. Adversarial Objectives

In this part, we review the related adversarial objectives as well as some approaches which are similar in spirit to our adversarial objectives.

Cost-Based Objectives. Our generic cost-bounded objective is not the only possible approach to model attacks in tabular domains. For example, works on adversarial robustness in

the context of decision tree-based classifiers often use per-feature constraints as adversarial constraints [53–55]. At the low level, these constraints are formalized either as bounds on L_∞ distance [54, 55], or using functions determining constraints for each feature value [53]. In these approaches, the feature constraints are independent. Such independence simplifies the problem. For example, the usage of L_∞ constraints enables to split a multidimensional optimization problem into a combination of simple one-dimension tasks [54], or to limit the set of points affected by the split change [53]. Unfortunately, per-feature constraints cannot realistically capture the *total cost* of mounting an attack: the aggregate cost of all the feature modifications required to produce an adversarial example, which is crucial to capture in tabular domains.

Also related to our cost-based proposal, Pierazzi et al. [29] introduce a general framework for defining attack constraints in the *problem space*. Our cost-based objective can be thought as an instance of this framework: we encode the problem-space constraints in the set of feasible examples.

Our cost model resembles the Gower distance [56], which is also a sum of "dissimilarities" across different categories of features. As opposed to this distance, our cost model can accommodate a wider class of numeric features, e.g., with a non-linear cost of changes. Also, it is not bounded to $[0, 1]$ interval providing flexibility to model a wider range of applications.

Utility-Based Objectives. The literature on *strategic classification* also considers utility-oriented objectives [57–59] for their agents. In this body of work, however, agents are not considered adversaries, and the gain is typically limited to

12

$\{+1, -1\}$ reflecting the classifier decision. Our model supports arbitrary gain values, which enables us to model broader interests of the adversary such as revenue. Only the work by Sundaram et al. [60] supports gains different from $+1$ or -1, but they focus on PAC-learning guarantees in the case of linear classifiers, whereas our goal is to provide practical attack and defense algorithms for a wider family of classifiers.

B. Attack Strategies

Tabular Domains. Several works have proposed attacks on tabular data. Ballet et al. [51] and Cartella et al. [61] propose to apply existing continuous attacks to tabular datasets. The authors focus on crafting imperceptible adversarial examples using standard methods from the image domain. They adapt these methods such that less "important" features (low correlation with the target variable) can be perturbed to a higher degree than other features. This corresponds to a special case within our framework, in which the feature-modification costs depend on the feature importance, with the difference that these approaches cannot guarantee that the proposed example will be feasible. Levy et al. [62] propose to construct a surrogate model capable of mimicking the target classifier. A part of this surrogate model is a feature-embedding function which maps tabular data points to a homogeneous continuous domain. They apply projected gradient descent to produce adversarial examples in the embedding space and map the resulting examples back to the tabular domain. As opposed to our methods, Levy et al. cannot provide any guarantee that the produced adversarial example lay in the feasible set. Finally, Kantchelian et al. [63] propose a MILP-based attack and its relaxation within different L_p cost models against random-forest models. Our attack differs from these three methods as they use L_p or similar bounds, whereas we use a cost bound that can capture realistic constraints as explained in Sections I and III.

Text Domains. Our universal greedy attack algorithm is similar to the methods for attacking classifiers that operate on text [14–19, 22]. All these works, however, make use of adversarial constraints such as restrictions on the number of modified words or sentences. These constraints do not apply to tabular domains, as simply considering "number of changes" does not address the heterogeneity of features. Our algorithms also differ from these approaches in that we incorporate complex adversarial costs in the design of the algorithms. For example, the Greedy attack by Yang et al. [14] uses the target classifier's confidence for choosing the best modifications to create adversarial examples while accounting for the number of modifications. Our framework not only considers the number of modifications but also their cost, thus capturing richer constraints of the adversary.

C. Adversarial Training

We discuss existing defense methods and techniques with related goals, which appear in the context of decision tree-based models. Adversarial robustness of such classifiers has been studied extensively [53–55, 64, 65]. These works assume independent per-feature adversarial constraints, e.g., based on the L_∞ metric. Our adversarial models, and thus our attacks and defenses, are capable of capturing a broader class of adversarial cost functions that depend on feature modifications and better model the adversary's constraints as we explain in Section VII-A.

VIII. Concluding Remarks and Future Work

In this paper, we have revisited the problem of adversarial robustness when the target machine-learning model operates on tabular data. We showed that previous approaches, tailored to produce adversarial image or text examples, and defend from them, perform poorly when used in tabular domains. This is because they are conceived within a threat model that does not capture the capabilities and goals of the tabular adversaries.

We introduced a new framework to design attacks and defenses that account for the constraints existing in tabular adversarial scenarios: adversaries are limited by a budget to modify features, and adversaries can assign different utility to different examples. Having evaluated these attacks and defenses on three realistic datasets, we showed that our novel utility-based defense not only generates models which are robust against utility-aware adversaries, but also against cost-bounded adversaries. On the contrary, performing adversarial training considering a cost-bounded adversary—as traditionally done in the literature—is a poor defense against adversaries focused on utility in some scenarios.

Although our high-level cost and utility-based framework is designed for tabular data, it can be applied to any case where possible adversarial actions can be modeled using generic costs, and where different attacks can have a different value for the adversary. As an example, our adversarial model could be applied to defending against attacks on a visual traffic sign recognition [66], where the attack harms could vary wildly from the misclassification of one type of road sign to another.

The main limitation of our work is that the effectiveness of the defenses relies on strong assumptions. The defender has to not only correctly model the adversary's objective, but also to estimate the adversary's cost model, budget, and how much value they ascribe to the success of each attack. More work is needed to understand the implications of misspecifications of this model, and to design algorithms that can provide protection even under misspecifications.

References

[1] Salah Ghamizi, Maxime Cordy, Martin Gubri, Mike Papadakis, Andrey Boytsov, Yves Le Traon, and Anne Goujon. Search-based adversarial testing and improvement of constrained credit scoring systems. In *ESEC/FSE*, 2020.

[2] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. Yes, machine learning can be more secure! A case study on android malware detection. *CoRR*, 2017.

[3] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, 2017.

[4] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. Adversarial malware binaries: Evading deep learning for malware detection in executables. *CoRR*, 2018.

[5] Jinyuan Jia and Neil Zhenqiang Gong. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *USENIX*, 2018.

[6] Bogdan Kulynych, Rebekah Overdorf, Carmela Troncoso, and Seda F. Gürses. POTs: protective optimization technologies. In *FAT**, 2020.

[7] Kendra Albert, Jonathon Penney, Bruce Schneier, and Ram Shankar Siva Kumar. Politics of adversarial machine learning. *CoRR*, 2020.

[8] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, 2013.

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, 2014.

[10] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Euro S&P*, 2016.

[11] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, 2016.

[12] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *S&P*, 2017.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, 2017.

[14] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. Greedy attack and Gumbel attack: Generating adversarial examples for discrete data. *JMLR*, 2020.

[15] Yutong Wang, Yufei Han, Hongyan Bao, Yun Shen, Fenglong Ma, Jin Li, and Xiangliang Zhang. Attackability characterization of adversarial evasion attack on discrete data. In *KDD*, 2020.

[16] Boxin Wang, Hengzhi Pei, Han Liu, and Bo Li. Adv-Codec: Towards a unified framework for adversarial text generation. *CoRR*.

[17] Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexandros G Dimakis, Inderjit S Dhillon, and Michael Witbrock. Discrete adversarial attacks and submodular optimization with applications to text classification. *CoRR*, 2018.

[18] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *ACL*, 2018.

[19] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *IJCAI*, 2018.

[20] Michele Carminati, Luca Santini, Mario Polino, and Stefano Zanero. Evasion attacks against banking fraud detection systems. In *RAID*, 2020.

[21] Mahmood Sharif, Lujo Bauer, and Michael K. Reiter. On the suitability of l_p-norms for creating and preventing adversarial examples. *CoRR*, 2018.

[22] Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep learning models in natural language processing: A survey.

[23] Xiao Zhang and David Evans. Cost-sensitive robustness against adversarial examples. *CoRR*, 2018.

[24] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.

[25] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. *CoRR*, 2018.

[26] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NeurIPS*, 2019.

[27] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 2018.

[28] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Towards the science of security and privacy in machine learning. *CoRR*, 2016.

[29] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ML attacks in the problem space. *IEEE S&P*, 2020.

[30] Daniel Lowd and Christopher Meek. Adversarial learning. In *KDD*, 2005.

[31] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Asia-CCS*, 2006.

[32] Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. In *ICML*, 2019.

[33] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *ICLR*, 2021.

[34] Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. On the effectiveness of adversarial training against common corruptions. *CoRR*, 2021.

[35] Bogdan Kulynych, Jamie Hayes, Nikita Samarin, and Carmela Troncoso. Evading classifiers in discrete domains with provable optimality guarantees. *CoRR*, 2018.

[36] Roni Stern, Rami Puzis, and Ariel Felner. Potential search: A bounded-cost search algorithm. In *ICAPS*, 2011.

[37] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Sys. Sci. and Cybernetics*, 1968.

[38] Roni Stern, Ariel Felner, Jur van den Berg, Rami Puzis, Rajat Shah, and Ken Goldberg. Potential-based bounded-cost search and anytime non-parametric A*. *Artificial Intelligence*, 2014.

[39] Richard E. Korf. Iterative-Deepening-A*: An optimal admissible tree search. In *Joint Conference on Artificial Intelligence*, 1985.

[40] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *J. ACM*, 1985.

[41] Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 1999.

[42] Laurence A. Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Math. Oper. Res.*, 1982.

[43] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

[44] Konstantinos Slavakis, Yannis Kopsinis, and Sergios Theodoridis. Adaptive algorithm for sparse system identification using projections onto weighted l1 balls. In *ICASSP*, 2010.

[45] Guillaume Perez, Sebastian Ament, Carla Gomes, and Michel Barlaud. Efficient projection algorithms onto the weighted l1 ball. *CoRR*, 2020.

[46] Zafar Gilani, Ekaterina Kochmar, and Jon Crowcroft. Classification of twitter accounts into automated agents and human users. In *ASONAM*, 2017.

[47] Kaggle. IEEE-CIS fraud detection, 2019. URL https://www.kaggle.com/c/ieee-fraud-detection.

[48] Kaggle. Home credit default risk, 2019. URL https://www.kaggle.com/c/home-credit-default-risk.

[49] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, 2021.

[50] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artif. Intell.*, 1970.

[51] Vincent Ballet, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, Marcin Detyniecki, et al. Imperceptible adversarial attacks on tabular data. In *Robust AI in FS NeurIPS Workshop*, 2019.

[52] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *ICML*, 2020.

[53] Yizheng Chen, Shiqi Wang, Weifan Jiang, Asaf Cidon, and Suman Jana. Cost-aware robust tree ensembles for security applications. In *USENIX*, 2021.

[54] Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. 2019.

[55] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. In *ICML*, 2019.

[56] John C Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 1971.

[57] Moritz Hardt, Nimrod Megiddo, Christos H. Papadimitriou, and Mary Wootters. Strategic classification. In *ITCS*, 2016.

[58] Jinshuo Dong, Aaron Roth, Zachary Schutzman, Bo Waggoner, and Zhiwei Steven Wu. Strategic classification from revealed preferences. In *EC*, 2018.

[59] Smitha Milli, John Miller, Anca D Dragan, and Moritz Hardt. The social cost of strategic classification. In *FAT**, 2019.

[60] Ravi Sundaram, Anil Vullikanti, Haifeng Xu, and Fan Yao. Pac-learning for strategic classification. In Marina Meila and Tong Zhang, editors, *ICML*, 2021.

[61] Francesco Cartella, Orlando Anunciacao, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. Adversarial attacks for tabular data: Application to fraud detection and imbalanced data. *SafeAI Workshop at AAAI*, 2021.

[62] Eden Levy, Yael Mathov, Ziv Katzir, Asaf Shabtai, and Yuval Elovici. Not all datasets are born equal: On heterogeneous data and adversarial examples. *CoRR*, 2020.

[63] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. Evasion and hardening of tree ensemble classifiers. In *ICML*, 2016.

[64] Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. Treant: training evasion-aware decision trees. *Data Min. Knowl. Discov.*, 2020.

[65] Daniël Vos and Sicco Verwer. Efficient training of robust decision trees against adversarial examples. In Marina Meila and Tong Zhang, editors, *ICML*, 2021.

[66] Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. Darts: Deceiving autonomous cars with toxic signs. *CoRR*, 2018.

APPENDIX A
OTHER POSSIBLE ADVERSARIAL OBJECTIVES

We propose a cost-oriented and a utility-oriented adversarial objective in Section C-B. These are not the only possible formalizations for our high-level goals. One other approach is an adversary maximizing utility subject to a cost budget:

$$\max_{x \in \mathcal{F}(x,y)} \mathbf{1}[F(x) \neq y] \cdot u_{x,y}(x')$$
$$= \mathbf{1}[F(x) \neq y] \cdot [g(x') - c(x,x')]_+ \quad (19)$$
$$\text{s.t. } c(x,x') < \varepsilon$$

This formalization is a middle ground between our cost-constrained and utility-constrained objectives: On the one hand, the adversary is aware of the utility of a given example. On the other hand, they do not adjust their budget for different examples, i.e. the constraint for $10 and $1,000 stays the same, even though the adversary clearly differentiates in their value. We conducted preliminary experiments with this objective, and

Algorithm 2 Cost-Ball Projection Algorithm

> **Input** $\overline{x}, \overline{x}', c, \varepsilon, \mathcal{C}, \mathcal{I}$
> **Output** $\delta^* = P_{\tilde{B}_\varepsilon(x,y)}(\overline{x}')$

1: $\delta = \overline{x}' - \overline{x}$
2: $w_i := \begin{cases} \min_{t \in \mathcal{F}_i(x,y)} c_i(x_i, t), & \text{if } i \in \mathcal{C} \\ c_{j-}(x), & \text{if } i \in \mathcal{I} \text{ and } \delta_i < 0 \\ c_{j+}(x), & \text{if } i \in \mathcal{I} \text{ and } \delta_i \geq 0 \end{cases}$
3: $z_i := \frac{\delta_i}{w_i}$
4: $\pi_z() := \text{Permutation} \uparrow (z)$
5: $z_i := z_{\pi_z(i)}$
6: $J := \max \left\{ j : \frac{-\varepsilon + \sum_{i=j+1}^{m} w_{\pi_z(i)} \delta_{\pi_z(i)}}{\sum_{i=j+1}^{m} w_{\pi_z(i)}^2} > z_j \right\}$
7: $\lambda := \frac{-\varepsilon + \sum_{j=J+1}^{m} w_{\pi_z(j)} \delta_{\pi_z(j)}}{\sum_{j=J+1}^{m} w_{\pi_z(j)}^2}$
8: $\delta_i^* := \text{sign}(\delta_i) \max(\delta_i - w_i \lambda, 0)$

> **Return** δ_i^*

The highlighted parts indicate the differences with respect to the sort-based weighted L_1 projection algorithm [45]. The function $\pi_z(i)$ denotes an outcome of permutation. Permutation $\uparrow (z)$ is a sort permutation in an ascending order.

its results are marginally different from the cost-bounded one in our experimental setup.

APPENDIX B
DETAILS ON ADVERSARIAL TRAINING

We describe our modifications to the traditional adversarial training pipeline. Our training procedure is a version of the well-known adversarial training algorithm based on the PGD method [13]. We design an adapted projection algorithm to solve Eq. (14), presented in Algorithm 2. This algorithm is an extension of an existing sort-based weighted L_1 projection algorithm [44, 45]. It takes as input a sample $\overline{x}$ and a perturbed sample $\overline{x}'$, and returns a valid perturbation vector δ such that $\overline{x} + \delta$ lies within the cost budget. With respect to the algorithm by Perez et al. [45], we introduce the capability to assign different weights based on a feature type and perturbation sign (line 2, in blue) to support our cost function in Eq. (13). The full proof of correctness for this algorithm is described in the extended version of the paper.

APPENDIX C
ADDITIONAL DETAILS ON THE EXPERIMENTS

We provide the details of our experimental setup.

A. Hyperparameter selection

We list our defence and attack parameters in Table III. TabNet parameters are denoted according to the original paper [49]. We set the virtual batch size to 512. As training the clean baseline for HomeCredit was prone to overfitting in our setup, we reduced the number of training epochs to 100. Other hyperparameters were selected with a grid search.

Algorithm 3 PGD-Based Attack

Input: P, initial example x, label y, costs w, cost bound ε.
Output: Adversarial example x^*

1: $\alpha := 2\frac{\varepsilon}{n}$
2: $\delta := 0$
3: **for** j in $1..n$ **do**
4: $\quad \nabla := \nabla_\delta \ell(f_\theta(\phi(x) + \delta), y)$
5: $\quad \delta := \delta + \alpha \frac{\nabla}{\|\nabla\|_1}$
6: $\quad \delta := P_{\mathcal{B}_{w,\varepsilon}}(\delta)$
$\quad x^* = P_{\mathcal{F}}(\delta)$
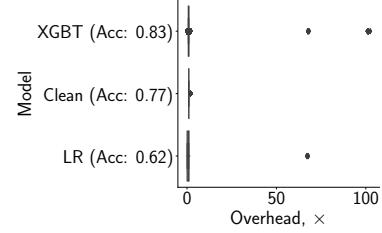
> **Return** x^*

Fig. 5: The distribution of cost overhead of adversarial examples obtained with UG over minimal-cost adversarial examples obtained with UCS on `IEEECIS`. Most UG adversarial examples have cost close to the minimal, although there exist outliers.

B. Dataset Processing and Adversarial Cost Models

For each dataset, we create an adversarial cost based on hypothetical scenarios. In this section, we describe how we process the data, and how we assign costs to modifications of the features in each dataset.

1) `TwitterBot`*:* We use 19 numeric features from this dataset. We drop three features for which we cannot compute the effect of a transformation as we do not have access to the original tweets. We use the number of followers as the adversary's gain. We assign costs of features based on estimated costs to purchase Twitter accounts of different characteristics on darknet markets.

2) `IEEECIS`*:* We ascribe cost of changes, assuming that the adversary can change the device type and email address at a small cost. The device type can be changed with low effort using specific software. Email domain can be changed with a registration of a new email address which typically cannot be automated. Although also low cost, it takes more time and effort than changing the device time. We reflect these assumptions ascribing the costs \$0.1 and \$0.2 to these changes. Changing the type of the payment card requires obtaining a new card, which costs approximately \$20 in US-based darknet marketplaces as of the time of writing. We consider the transaction amount as a gain obtained by an adversary.

3) `HomeCredit`*:* The main goal of the adversary in this task is receiving a credit approval. As one example represents a loan application, we set the credit amount to be the gain of the example. All features which can be used by an adversary

TABLE III: IEEECIS and HomeCredit attack and defense parameters

Parameter	Value range
Adversarial Training (IEEECIS)	
Batch size	2048
Number of epochs	400
PGD iteration number	20
TabNet hyperparameters	$N_D = 16, N_A = 16, N_{steps} = 4$
ε *(for CB models)*	$[1, 3, 10, 30]$
τ *(for UB models)*	$[0, 10, 20, 50, 100, 200, 500]$
Attacks (IEEECIS)	
Max. iterations	100K
ε *(for CB attacks)*	$[1, 3, 10, 30]$
τ *(for UB attacks)*	$[0, 10, 50, 500, 1000]$
Adversarial Training (HomeCredit)	
Batch size	2048
Num. of epochs	100
TabNet hyperparameters	$N_D = 16, N_A = 16, N_{steps} = 4$
Num. of PGD iterations	20
ε *(for CB models)*	$[1, 10, 100, 1000, 10000]$
τ *(for UB models)*	$[300K, 400K, 500K, 600K, 800K]$
Attacks (HomeCredit)	
Num. of iterations	100
ε *(for CB attacks)*	$[1, 10, 100, 1K, 10K]$
τ *(for UB attacks)*	$[10K, 300K, 400K, 500K, 600K, 800K]$

are listed in Table VI with the costs we ascribe to them. We assume six groups of features and estimate the cost as follows:

- *Group 1*: Features that an adversary can change with negligible effort such as an email address, weekday, or hour of the loan application. We ascribe $0.1 cost to these transformations.
- *Group 2*: Features associated to income. We use these as numerical features to illustrate the flexibility of our method. We assume that to increase income by $1, the adversary needs to pay $1.
- *Group 3*: Features associated to changing a phone number. Based on the US darknet marketplace prices as of the time of writing, we estimate that purchasing a SIM card costs $10.
- *Group 4*: Features related to official documents which can be temporally changed. For example, a car's ownership can be transferred from one person to another for the application period, and returned to the original owner after it. We ascribe a cost of $100 to these changes.
- *Group 5*: Features that require either document forging or permanent changes to a person's status. For instance, purchasing a fake university diploma. These are expensive changes, and we estimate their cost at $1,000.
- *Group 6*: Features related to credit scores provided by external credit-scoring agencies. We estimate the cost of changes in this group with a manipulation model based on a real-world phenomenon of credit piggybacking, described in the extended version of the paper.

TABLE IV: Costs of changing a feature in TwitterBot dataset

Feature	Estimated cost, $
likes_per_tweet	0.025
retweets_per_tweet	0.025
user_tweeted	2
user_replied	2

TABLE V: Costs of changing a feature in IEEECIS dataset

Feature	Estimated cost, $
DeviceType	0.1
P_emaildomain	0.2
card_type	20

TABLE VI: Costs of changing a feature in HomeCredit

Feature	Estimated cost, $
NAME_CONTRACT_TYPE	0.1
NAME_TYPE_SUITE	0.1
FLAG_EMAIL	0.1
WEEKDAY_APPR_PROCESS_START	0.1
HOUR_APPR_PROCESS_START	0.1
AMT_INCOME_TOTAL	1
FLAG_EMP_PHONE	10
FLAG_WORK_PHONE	10
FLAG_CONT_MOBILE	10
FLAG_MOBIL	10
FLAG_OWN_CAR	100
FLAG_OWN_REALTY	100
REG_REGION_NOT_LIVE_REGION	100
REG_REGION_NOT_WORK_REGION	100
LIVE_REGION_NOT_WORK_REGION	100
REG_CITY_NOT_LIVE_CITY	100
REG_CITY_NOT_WORK_CITY	100
LIVE_CITY_NOT_WORK_CITY	100
NAME_INCOME_TYPE	100
CLUSTER_DAYS_EMPLOYED	100
NAME_HOUSING_TYPE	100
OCCUPATION_TYPE	100
ORGANIZATION_TYPE	100
NAME_EDUCATION_TYPE	1000
NAME_FAMILY_STATUS	1000
HAS_CHILDREN	1000

TABLE VII: Effect of beam size B in the Universal Greedy algorithm on the IEEECIS dataset. The success rates are close for all choices of the beam size, thus the beam size of one offers the best performance in terms of runtime.

Cost bound → Beam size ↓	Adv. success, % 10	30	Gain	∞
1	45.32	**56.57**	**56.22**	**68.20**
10	45.32	56.01	55.65	56.01
100	45.32	56.53	56.18	56.53

Cost bound → Beam size ↓	Success/time ratio 10	30	Gain	∞
1	**3.78**	**4.80**	**2.53**	**2.06**
10	2.14	2.25	1.31	1.15
100	0.66	0.65	0.65	0.66

TABLE VIII: Summary of related work in terms of three aspects: adversarial models, attack strategies, and defense strategies. Adversarial models: *Adv. cost* — description of (an equivalent of) an adversarial cost model. *Adv. utility* — whether adversarial gain is incorporated into the model. Attacks: *Targets* — which target models can be attacked. *Feasibility* — whether the attack is guaranteed to produce a feasible adversarial example. *Algorithm* — a short description of the algorithmic approach. defenses: *Arch.* — which model architectures are supported.

	Adversarial models		Attacks			Defenses
Adversarial Model	Adv. cost	Adv. utility	Targets	Feasibility	Algorithm	Arch.
Ballet et al. [51]	Feature-importance based	—	Differentiable	✗	Gradient-based	—
Cartella et al. [61]	Feature-importance based	—	**Any**	✗	ZOO	—
Levy et al. [62]	Distance based	—	**Any**	✗	Gradient-based	—
Kantchelian et al. [63]	L_p	—	Tree-based	✓	MILP	—
Andriushchenko and Hein [54]	L_∞	—	Tree-based	✓	Custom	Tree-based
Chen et al. [53]	Per-feature constraints	—	—	—	—	Tree-based
Calzavara et al. [64]	Per-feature constraints	—	Tree-based	✓	Exhaustive search	Tree-based
Vos and Verwer [65]	Per-feature constraints	—	—	—	—	Tree-based
Ours	**Generic** (Section IV-B)	✓	**Any**	✓	Graph search	Differentiable

(a) `IEEECIS`: ● CB-trained models ● UB-trained models × Clean model

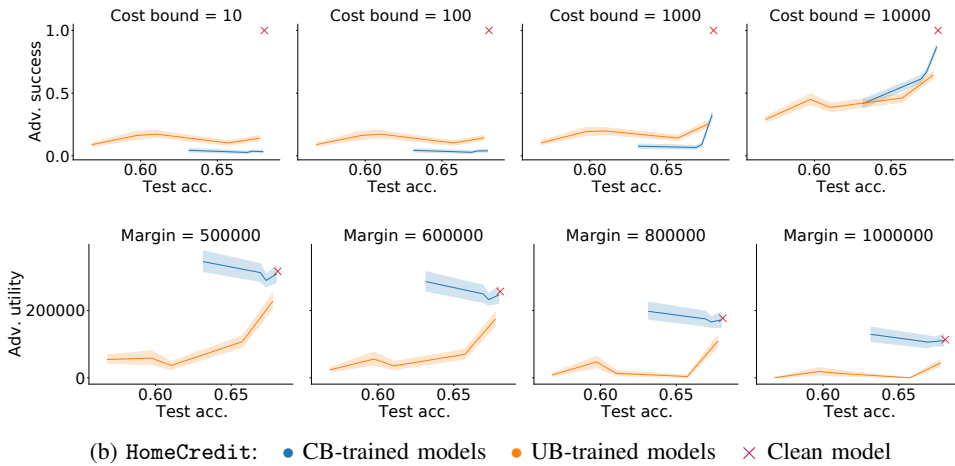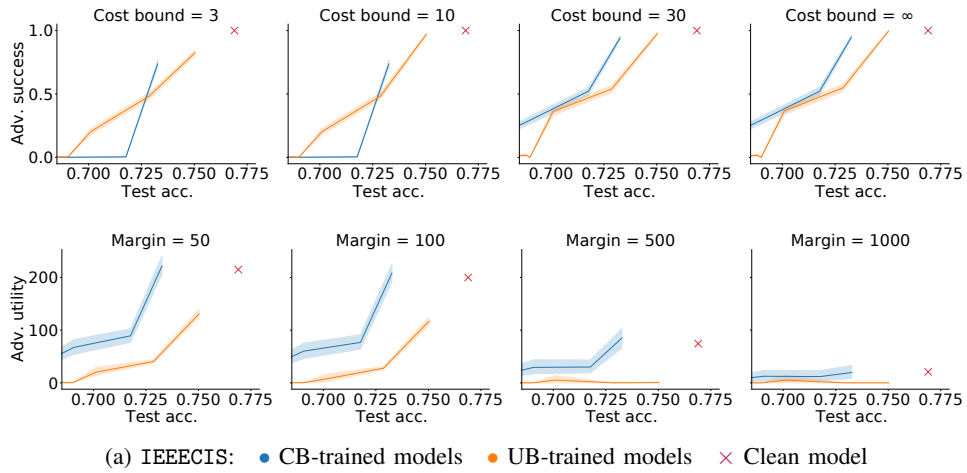(b) `HomeCredit`: ● CB-trained models ● UB-trained models × Clean model

Fig. 6: *Accuracy-robustness and utility-robustness trade-offs* for Cost-bounded and Utility-bounded adversarially trained models. The curves show accuracy (x-axis) and utility and success rate (x-axis) for the utility- and cost-bounded models presented in Fig. 3 and Fig. 4. When one curve is strictly below the other curve, it provides a better trade-off since it has better robustness for the same accuracy. Utility-bounded models consistently show better trade-offs for all utility-aware attacks. For CB attacks the situation is less consistent: for small cost-bounds CB defense outperforms utility-bounded one, while for the largest budgets utility bounded shows better results.