Identifying Insufficient Data Coverage for Ordinal Continuous-Valued Attributes

Abolfazl Asudeh
University of Illinois at Chicago
asudeh@uic.edu

Nima Shahbazi
University of Illinois at Chicago
nshahb3@uic.edu

Zhongjun Jin
University of Michigan
markjin@umich.edu

H. V. Jagadish
University of Michigan
jag@umich.edu

ABSTRACT

Appropriate training data is a requirement for building good machine-learned models. In this paper, we study the notion of *coverage* for ordinal and continuous-valued attributes, by formalizing the intuition that the learned model can accurately predict only at data points for which there are "enough" similar data points in the training data set.

We develop an efficient algorithm to identify uncovered regions in low-dimensional attribute feature space, by making a connection to Voronoi diagrams. We also develop a randomized approximation algorithm for use in high-dimensional attribute space. We evaluate our algorithms through extensive experiments on real datasets.

KEYWORDS

Responsible Data Science; Trustworthy AI; Fairness in Machine Learning; Bias Detection

ACM Reference Format:
Abolfazl Asudeh, Nima Shahbazi, Zhongjun Jin, and H. V. Jagadish. 2021. Identifying Insufficient Data Coverage for Ordinal Continuous-Valued Attributes. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3448016.3457315

1 INTRODUCTION

Machine learning (ML) and predictive analytics are shaping every corner of human life, from autonomous vehicles to healthcare, and even in predictive policing and data-driven sentencing. A critical question for the decision maker, especially in applications that may impact human life, is how much to trust the outcome of the model.

It is easy to see that the accuracy of ML model prediction depends on the data used for training: after all, the model learns the phenomena that training data represent. As a first step, we may desire that the data should represent the underlying data distribution from which the production data will be drawn. But that is not enough, since it only tells us about model performance in the aggregate. For any query point, what we want is that the training data should include enough examples similar to it. Otherwise, the model predicts the query point according to points not similar to the query. If the "behavior" of the query point is similar to the training data, the model (luckily) may provide a good outcome. Even so, the model outcome is not trustworthy in these situations, at least for critical decision making.

To consider one recent example in the news, Amazon developed software to screen employment applications based on predicted success. Because the training data set had so few successful women in it, the program would downgrade women in its predictions [23]. While the model may have worked well for male applicants, it was clearly unsuited for women applicants, and was ultimately not used by Amazon in production. Similarly, using a real dataset of criminals [1], an ML model with more than 75% overall accuracy had a performance worse than random guess for Hispanic Females (because the training data did not contain enough samples from this category) [13]. In short, given a query point, we may not trust the output (such as prediction or classification) of an ML model about this point only if there are not enough points in the training data set for the ML model that are "similar" to the query point. This concept has been known as *coverage* in the literature [13, 38, 45, 47].

Poor coverage does not necessarily imply poor models. In a classification setting, for instance, having poor coverage in regions far from the boundary is likely to be immaterial since those points may not contribute to refining the boundary. However, as we show experimentally in § 6.2, poor coverage creates a risk that there may be a poor model. More specific analyses may be able to assess this risk, given additional information. However, the results become specific to the type of model being considered. Our goal here is to investigate datasets independent of the type of model being constructed. Furthermore, we note that poor model performance could be due to reasons (e.g. poorly labeled data for a minority group) other than lack of coverage.

Existing work is limited to categorical attributes with low-cardinality. In this paper, we extend this concept to the more challenging context of continuous-valued attributes. Returning to the example of Amazon's model for scoring job applicants, the detected bias was on a simple low-cardinality discrete attribute: sex. It is possible that the model also discriminates based on age, since most tech workers, and job applicants, are young. Simple solutions like binning age into "young" and "old" can lead to coarse groupings that are sensitive to thresholds chosen. For example, it may be inappropriate to treat a 35-yo as young but a 36-yo as old.

In short, we may not trust the result of a given query if the query point is not covered by the training data. One can make a pass over the training data to check if the given query point is covered. But this requires (i) access to the full training data set and (ii) the time to verify the coverage based on it, neither of which may be feasible. Therefore, our challenge is to mark out in advance, for

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3457315

every possible query point, whether it is covered. We propose to address this challenge by identifying, in advance, the uncovered region in the feature-vector space, and to encode these in a manner that makes it inexpensive to determine which region any query point is in.

Identifying uncovered region up-front enables crucial benefits and two action items. First, uncovered region shows potential deficiencies in the (training) data set. *Annotating* a data set with coverage information informs the data scientist about the uncovered regions when the model is being constructed as a signal to investigate the fitness of data for the model [58, 61]. Second, at query time, it generates a warning that the outcome might not be trustworthy when a point is queried in an uncovered region. Whether to consider the outcome and how to take action is left to the model user.

Summary of contributions. In summary, our contributions in this paper are as follows:

- We formally define the notions of coverage over ordinal and continuous-valued attributes. Based on this, we define the problem of identifying uncovered regions of a given dataset. (§ 2)
- Making a connection to the Voronoi diagrams [14], we design an efficient algorithm for identifying uncovered regions for the two-dimensional case. We also provide a logarithmic search algorithm for checking if a query point belongs to an uncovered region. (§ 3)
- The problem search space size explodes as the number of dimensions increases. For higher dimensions, we design a randomized approximation algorithm based on the geometric notion of ε-**net**. We first provide a theoretical upper-bound on the number of required samples for the adversarial case, which is exponential to the number of dimensions. Fortunately, it turns out in practice that the actual number of samples required for identifying the uncovered region is significantly smaller than the provided theoretical upper-bound. We design an algorithm based on exponential-search to identify the right number of samples for a given setting. We study how to report the uncovered region in multi-dimensional cases to the user. (§ 4)
- Finally, we conduct extensive experiments on real datasets to evaluate our proposed algorithms. (§ 6)

2 PROBLEM DEFINITION

2.1 Data Model

Consider a dataset $\mathcal{D}$ with n tuples, each consisting of d ordinal attributes $X = \{x_1, x_2, \cdots, x_d\}$. Attribute values may be discrete ordinal (e.g. population) or continuous-valued (e.g. height). In either case, we normalize attribute values to lie in the range $[0, 1]$, with values drawn from the set of rational or real numbers. For every tuple $t \in \mathcal{D}$, we use the notation $t[i]$ to show the value of t on attribute $x_i \in X$. In practice, the data scientist may be interested to study coverage over a subset of attributes, called "attributes of interest". In such cases, we assume X is the set of attributes of interest. The dataset also contains target attributes $Y = \{y_1, \cdots, y_{d'}\}$ that are not considered for the coverage problem.

2.2 Coverage Model

Given a query point $q \in [0, 1]^d$, where $q[i]$ shows the value of q with regard to $x_i \in X$, the dataset $\mathcal{D}$ is not representative for q, if there are not "enough" data points in $\mathcal{D}$ that are "similar" to q. In such cases, we say q is not covered by $\mathcal{D}$. We use the simplest possible interpretations for "similar" and "enough" to obtain the following formal definition:

Definition 1 (Coverage of a Point). *Given a dataset $\mathcal{D}$ with d attributes $X = \{x_1, x_2, \cdots, x_d\}$, query point $q \in [0, 1]^d$, a distance function $\Delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, a vicinity value ρ, and a threshold value k, the coverage of q by $\mathcal{D}$ is verified as follows:*

$$Cov_{\rho,k}(q, \mathcal{D}) = \begin{cases} true & if\ |\{t \in \mathcal{D} \mid \Delta(t, q) \leq \rho\}| \geq k \\ false & otherwise \end{cases}$$

In the rest of the paper, we simplify the notation to $Cov(q, \mathcal{D})$ when ρ and k are clear by the context. Also, in the rest of the paper use ℓ_2 norm as the distance measure.

We rely on the domain knowledge of the data scientists to specify the vicinity and threshold, since these values are application specific and may vary by context. One can use techniques we shall propose in § 5, for tuning the parameters. We also evaluate experimentally the impact of these parameter choices in § 6. Using Definition 1, we now define the uncovered regions as follows.

Definition 2 (Uncovered Region). *Given a dataset $\mathcal{D}$ with d attributes $X = \{x_1, x_2, \cdots, x_d\}$, a distance function $\Delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, a vicinity value ρ, and a threshold value k, the uncovered region $\mathcal{U}$ is the set of points (value combinations) that are not covered by $\mathcal{D}$. Formally, $\mathcal{U} = \{q \in [0, 1]^d \mid Cov(q, \mathcal{D}) = false\}$.*

Figure 1(a) shows a sample dataset with two attributes x_1 and x_2. Every tuple t in the dataset is shown as a black dot in the figure. For each tuple $t \in \mathcal{D}$, let, $\circ_t$ be the circle with radius ρ centered at it, as drawn in the figure. Note that any point within the circle $\circ_t$ has the distance less than ρ from t. As a result, every point in the space that falls within less than k circles contains less than k tuples in its neighborhood and, according to Definition 1, is uncovered.

Setting k to be 2, the uncovered region in the example of Figure 1(a) is colored in red. As shown in the figure, the uncovered region is a collection of disjoint shapes, not convex in general, and with borders identified by arcs, each belonging to the proximity circle of a tuple.

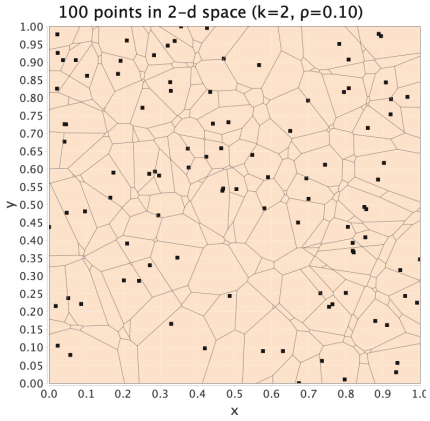
2.3 Problem Formulation

Our challenge is to address two tasks: first, given a data set, to find the uncovered region; and second, given a query point, to determine whether it is in an uncovered region. Formally, we define the following problems:

Problem 1 (Uncovered Region Discovery). *Given dataset $\mathcal{D}$ with n tuples over d attributes, and the values ρ and k, identify the uncovered region $\mathcal{U}$ as defined in Definition 2.*

Problem 2 (Uncovered Query Answering). *Given dataset $\mathcal{D}$, the values ρ and k, and a query point q check if q is uncovered. That is, to identify if $|\{t \in \mathcal{D} \mid \Delta(t, q) \leq \rho\}| < k$.*

Note that Problems 2 and 1 are related. In particular, identifying the uncovered region in Problem 1, enables answering Problem 2 by



(a) uncovered region (red), $k = 2$ (b) 1st order voronoi diagram (c) 2nd order voronoi diagram

Figure 1: Illustration of 100 random points in range $[0, 1] \times [0, 1]$. (a) For every tuple t, $\circ_t$ is the circle centered at t.

checking if the query point belongs to the identified region. However, one might be able to answer Problem 2 without identifying the uncovered region first. Similarly, an efficient solution for Problem 2 enables approximately answering Problem 1 by traversing through a discretized search space.

Lack of coverage is only one of several signals that are important in determining the trustworthiness of an analysis outcome based on the data. It is informative when considered along with other factors [7] such as basic performance, robustness [43] or stability [9] of outcome, and explainability [52]. For example, one may use the consistency of the outcomes of multiple model ensembles as a signal for trust. As such, we seek only to provide coverage information, whether by identifying uncovered regions in a data set or by flagging poor coverage for a specific query, as an indicator to be used by the data scientist or data user in conjunction with other information and in the context of the particular application.

In the following, we provide algorithms for Problems 1 and 2 for the two-dimensional ($d = 2$) case and the general multi-dimensional case, respectively. Before moving to the technical details though, we would like to provide a high-level discussion about the output complexity of the uncovered region that, setting aside the complexity of the discovery algorithm, may be challenging. As discussed earlier and observed in Figure 1(a), the uncovered region can be expressed by its border, the union of d-dimensional arcs each belonging to the proximity circle of a tuple. That is, complexity of the region is determined by the number of arcs contributing to the border of the region. In following sections, we shall use computational geometry notions Voronoi diagrams (Lemma 1) and VC-dimensionality (Theorem 3) to bound the output complexity of the uncovered region. In particular, following Lemma 1, the number of arcs at the border of uncovered region in 2D are at most k times more than the number of k-voronoi cells.

3 TWO DIMENSIONAL COVERAGE

In this section, we study the two dimensional case where $X = \{x_1, x_2\}$. Figure 1(a) shows an example of such case. As discussed in the previous section, we use $\circ_t$ to represent the circle with radius ρ around the tuple t. For every query point q, it is easy to observe that it is uncovered, i.e. $Cov(q, \mathcal{D})$ =*false*, iff $|\{t \in \mathcal{D} \mid q \text{ is inside } \circ_t\}| < k$. Using this observation, the brute-force approach for finding the uncovered region draws the circles $\circ_t$ for each tuple $t \in \mathcal{D}$ and considers the partitioning of the space by the arrangement [28] of the circles. For each cell in the partition, the algorithm counts the number of circles inside which the partition falls in. If the number is less than k, it marks the cell as uncovered. For example, in Figure 1(a) the circle of each tuple is drawn around it. The intersections of the circles partition the space to different cells. Every cell is the intersection of multiple circles. Any point inside a cell is in ρ-vicinity of its circle. The algorithm marks as uncovered any cell that is the intersection of less than k circles – painted in red in the figure. This algorithm, however, is not practical. First, constructing the arrangement of circles on the space is not straightforward, let alone easy to implement efficiently. Second, the quadratic complexity of the number of cells in the arrangement of circles makes the algorithm inefficient. Finally, it is not clear how to organize and index the cells to later answer coverage queries (Problem 2) efficiently.

Therefore, in the rest of this section, our aim is to design efficient algorithms for Problems 1 and 2 in 2D. To do so, we make a key connection to the well-studied concept of voronoi diagrams [14–16, 29–31].

3.1 Review: k-th Order Voronoi Diagrams

Consider a dataset $\mathcal{D}$ in $\mathbb{R}^2$. For a pair of tuples t_i and t_j, let $h(t_i, t_j)$ be the line $h(i, j) = \{q \mid \Delta(q, t_i) = \Delta(q, t_j)\}$. Also, let the half-space $h^+(i, j)$ be the side of the line the includes t_i, i.e. $h^+(i, j) = \{q \mid \Delta(q, t_i) \leq \Delta(q, t_j)\}$. The locus of the points closer to t_i than any other points in $\mathcal{D}$, denoted by $\mathcal{V}(i) = \bigcap_{\forall i \neq j} h^+(i, j)$. The convex polygon $\mathcal{V}(i)$ is called the *voronoi cell* of tuple t_i. Following this, the collection of voronoi cells for all tuples in $\mathcal{D}$ is called the (first order) voronoi diagram of the dataset. Figure 1(b) shows the voronoi diagram for sample points in Figure 1. A (first order) voronoi diagram contains n cells, simply because there exists only

one cell per each tuple. Also, it has been shown that the diagram contains at most $3n-6$ edges and $2n-5$ vertices [29].

Higher order voronoi diagrams [4, 20, 21, 42] extend the voronoi diagrams from nearest neighbor to k-nearest neighbors. A k-voronoi cell $\mathcal{V}(S)$ is associated with a set $S \subseteq \mathcal{D}$ where $|S| = k$ and the k nearest neighbors of any point in $\mathcal{V}(S)$ are S. Formally, $\mathcal{V}(S) = \{q \mid \forall t_i \in S, t_j \in \mathcal{D}\backslash S : \Delta(q,t_i) \leq \Delta(q,t_j)\}$. The k-th order voronoi diagram (denoted as $V_k(\mathcal{D})$) is the collection of all k-voronoi cells of $\mathcal{D}$. Formally, $V_k(\mathcal{D}) = \{\mathcal{V}(S) : S \subseteq \mathcal{D}, |S| = k\}$. An example of 2nd order voronoi diagram is presented in Figure 1(c). The k-th order voronoi diagram of a dataset $\mathcal{D}$ in 2D contains $O(k(n-k))$ cells [20, 42]. In [42], D. T. Lee proposed an algorithm for finding the k-order voronoi diagrams that runs in $O(k^2 n \log(n))$ and has a space complexity of $O(k^2(n-k))$.

3.2 Discovering the Uncovered Region

We use the k-th order voronoi diagrams for discovering the uncovered region. Lemma 1 makes the necessary foundation for our algorithm design.

Lemma 1. *Consider a dataset $\mathcal{D}$ and its corresponding k-th order voronoi diagram. For every tuple $t \in \mathcal{D}$, let $\circ_t$ be the d-dimensional sphere (d-sphere) with radius ρ centered at t. Consider a k-voronoi cell $\mathcal{V}(S)$ in the k-th order voronoi diagram $V_k(\mathcal{D})$. Any point q inside the intersections of the d-spheres of tuples in S, i.e. $q \in \cap_{\forall t \in S} \circ_t$, is covered, while all other points in the region are uncovered. Formally:*

$$\forall q \in \mathcal{V}(S) : \; Cov_{\rho,k}(q, \mathcal{D}) = \begin{cases} true & if \; q \in \cap_{\forall t \in S} \circ_t \\ false & otherwise \end{cases}$$

Proof. Let R be the intersection $\cap \circ_t, \forall t \in S$. Every point falling inside a d-sphere $\circ_t$ has the distance of at most ρ from the corresponding tuple t. Therefore, the distance of any point in R is at most ρ from all tuples in S and since $|S| = k$, is covered. Now consider a point $q \in (\mathcal{V}(S) \setminus R)$. First, since q is in $\mathcal{V}(S)$, for any tuple t' that belongs to $(\mathcal{D} \setminus S)$, the distance of t' to q is not less than the maximum distance of S to q, i.e., $\Delta(q,t') \geq \max_{\forall t \in S} \Delta(q,t)$. Next, since $q \notin R$, there exists at least one tuple t in S such that $\Delta_M = \Delta(q,t) > \rho$. Following this, $\forall t' \in (\mathcal{D} \setminus S)$,

$$\Delta(q,t') \geq \Delta_M > \rho$$

Consequently, since $|S\backslash\{t\}| < k$, q is not covered. □

According to Lemma 1, for identifying the uncovered points in a region R of the k-th order voronoi diagram it is enough to only consider the tuples in S_R. We use this as the key observation for designing Algorithm 1 for discovering the uncovered region in the two dimensional case. The algorithm starts by constructing the k-th order voronoi diagram of the dataset, using [42] or any other efficient algorithm. Then for each voronoi cell $\mathcal{V}(S)$ in the diagram, it computes the intersection of the circles of the tuples in S and marks the portion of $\mathcal{V}(S)$ that falls outside it as uncovered. The algorithm's correctness follows Lemma 1. First, since the k-voronoi diagram partitions the space, every query point q should belong to one and only one cell. According to the lemma, if q is uncovered, it should fall outside the intersection of the corresponding tuples for the cell and hence discovered by Algorithm 1. After identifying the uncovered region, similar to Figure 1, we use the 2D map of $\{x_1, x_2\}$ value combinations to report the region to the user.

Algorithm 1 uncovered-2d

Input: dataset $\mathcal{D}$, k, ρ

```
1:  function Identify(D, k, ρ)
2:      V ← V_k(D); U ← {}
3:      for V(S) ∈ V do
4:          C ← ∩ ∘_t  ∀t ∈ S
5:          U.add(V(S)\C)
6:      return U, V
7:  function Report(U)
8:      map ← 2D map of [0,1] × [0,1]
9:      for ⟨x1, x2⟩ ∈map do
10:         if ⟨x1, x2⟩ ∈ U then color[x1, x2] ← red
11:         else color[x1, x2] ← green
12: function Query(q, V) // is q uncovered?
13:     find V(S) ∈ V such that q ∈ V(S)
14:     for t ∈ S do
15:         if Δ(t, q) > ρ then
16:             return true; //q is uncovered
17:     return false
```

Lemma 2. uncovered-2d-Identify *runs in $O(k^2 n \log n)$.*

Proof. Finding k-th order voronoi diagram is in $O(k^2 n \log n)$ [42]. For every cell in the diagram we take the intersection of k circles. Since the upper-bound on the number of cells in the diagram is $O(kn \log n)$ [4, 20, 42], this takes $O(k^2 n \log n)$. □

We would like to note that, following Lemma 1, since the number of k-voronoi cells in 2D is $O(k(n-k))$, the output complexity of uncovered region is bounded as $O(k^2(n-k))$. The minimal representation of the uncovered region, though, is achieved by the exact number of arcs defining the border of the region. We use the voronoi diagram in 2D because its complexity is linear in n and it enables a logarithmic query answering algorithm we shall explain next.

3.3 Query Answering

In addition to identifying uncovered region, we also rely on the rich literature of voronoi diagrams for efficiently checking if a query point is uncovered. In particular, instead of explicitly indexing the uncovered region, we only index the k-voronoi diagram during the preprocessing time. During the query time, the first step is to identify the voronoi cells that the query point q belongs to. This is done in $O(\log n)$ [4, 42]. Let $\mathcal{V}(S)$ be the k-voronoi cell q belongs to. Note that for every tuple $t' \in \mathcal{D}\backslash S$, $\Delta(t', q) \geq \max(\Delta(t,q), \forall t \in S)$. Hence, since $|S| = k$, if there exists a tuple $t \in S$ such that $\Delta(t,q) > \rho$, then q is uncovered. This step requires $O(k)$ operations. Consequently, the Query function of Algorithm 1 is in $O(k + \log n)$.

4 MULTI-DIMENSIONAL COVERAGE

So far in this paper, we considered the special 2D case where the coverage is studied over two attributes. In this section, we relax this assumption to the general case where $d \geq 2$.

The extension of the 2D case provides an exact algorithm for detecting the uncovered region: following Algorithm 1, construct the k-th order voronoi diagram in MD and, within every cell $\mathcal{V}(S)$, find the points outside the intersection of the hyper-circles of S. This algorithm can indeed be used effectively for low-dimensional spaces. Unfortunately, due to the curse of dimensionality, the exact algorithm described above is not practical when d, the number of attributes, is not a small constant. Therefore, in this section, we turn our attention to designing approximation algorithms.

4.1 Approximating the Uncovered Region

In this section we design a geometric approximation algorithm for identifying the uncovered regions in MD. But first, let us review some of the necessary concepts in the following.

Consider a range space $(\chi, \mathcal{R})$ where χ is the universe of points and $\mathcal{R}$ is the set of ranges, each containing a subset of points in χ. For example, χ can be $\mathbb{R}^2$ and $\mathcal{R}$ the set of disks in the plane. The VC-dimension of a range space is the maximum cardinality of $S \subseteq \chi$ shattered by $\mathcal{R}$. It means, for every set $s \subseteq S$, there exists a range $r \in \mathcal{R}$ that include s and no point from $S \backslash s$.

A set $N \subseteq \chi$ is an ε-**net** for χ, if for any range $r \in \mathcal{R}$, if $|r \cap \chi| > \varepsilon|\chi|$, then r contains at least one point of N (i.e., $|r \cap N| \neq \emptyset$). Let δ be a the VC-dimension of a range space $(\chi, \mathcal{R})$. For such a range space, an i.i.d. sample of size

$$O\Big(\frac{1}{\varepsilon}\big(\delta \log \frac{1}{\varepsilon} + \log \frac{1}{\phi}\big)\Big) \tag{1}$$

from the χ is an ε-**net** for the range space, with probability of at least $(1 - \phi)$ [34]. We use this for approximately *learning* the uncovered region.

Let net be an ε-**net** for the d dimensional query space $[0, 1]^d$. Let us label every sample point in the net as +1 if it is uncovered and -1 otherwise. Consider the region induced by the samples as the uncovered region; that is the (non-convex) hull $\mathcal{U}_a$ around the +1 points that does not include *any* of the -1 points. In other words, a classifier with no misclassification error on the samples. Now consider the exact uncovered region $\mathcal{U}$. Let $r = \mathcal{U} \oplus \mathcal{U}_a$ be the difference between $\mathcal{U}$ and $\mathcal{U}_a$, i.e., the set of points that are mis-classified as either covered or uncovered using $\mathcal{U}_a$. Clearly, any query point in r gets mis-classified by $\mathcal{U}_a$. Since net is an ε-**net**, if $|r \cap \chi| > \varepsilon|\chi|$, then r contains at least one point of net. In other words, the error induced by the approximation is bounded by ε.

In the following, we focus on the VC-dimension of the uncovered region, provide a theoretical upper-bound, and design a practical solution.

4.1.1 A theoretical upper-bound. The VC-dimension of the uncovered region play the critical role in constructing the ε-**net** for our algorithm. In the following, we first use our findings in § 3 and the available results on the complexity of voronoi diagrams to find an upper-bound in Theorem 3.

Theorem 3. *The VC-dimension of the uncovered region in $\mathbb{R}^d$ is bounded by $O\big((d+1)n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil + 1}\big)$.*

Proof. It is known that any finite sequence of combining (union, intersection, and complementing) range spaces with finite VC-dimension, results in a range space with finite VC-dimension [34]. We use this to find an upper-bound on the VC-dimension of the uncovered region. Following Lemma 1, for every k-voronoi cell $\mathcal{V}(S)$, the points inside the intersections of the spheres (with radius ρ) of tuples in S are covered and the points outside those belong to the uncovered region. As a result, for every k-voronoi cell, we can consider the covered points as a range that is the intersection of k d-spheres (d-dimensional spheres). The VC-dimension of a d-sphere is $d + 1$ [34]. For instance, VC-dimension of disks in the plane is 3, since it is the maximum number of points disks on the plane it can shatter. Therefore, the covered points inside a cell have the VC-dimension $k(d+1)$. The number of cells of a k-voronoi diagram is bounded by $O\big(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil}\big)$ [29]. Since the set of covered points is identified as the union of the covered points in each k-voronoi cells, the VC-dimension of the covered region is bounded by

$$k(d+1)\, O\big(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil}\big) = O\big((d+1) n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil + 1}\big)$$

The uncovered region is the complement of the covered region and has the same VC-dimension. □

Theorem 3 provides an upper-bound on the VC-dimension of the uncovered region. This, however, is an upper-bound for the adversarial cases that (as we shall experimentally observe) are far from practice. In our experiments samples of significantly smaller sizes are enough to create the ε-**net**. We first note that Theorem 3 uses the theoretical upper-bound the number of k-voronoi cells. Still, perhaps more importantly, it assumes that all voronoi cells contain some covered and uncovered points which seems very adversarial. In practice a substantial portion of the cells are either fully covered (resp. uncovered), i.e. any point in those regions are covered (resp. uncovered) and does not add to the complexity (VC-dimension) of the uncovered region. As a result, much smaller sample sizes are enough to build the ε-**net**. We shall experimentally evaluate this in § 6.

4.1.2 A practical solution. The upper-bound in Theorem 3 uses the number of k-voronoi cells to bound the VC-dimension. Alternatively we can use the complexity of the shape of the uncovered region for finding the VC-dimension. Looking at Figure 1(a), one can confirm that the border between the uncovered and covered region is described as the union of a set of d-dimensional arcs (d-arcs). Every d-arc is part of a d-sphere, having a VC-dimension of $d + 1$. Therefore, if the border of uncovered region is the union of N_{arc} d-arcs, the VC-dimension of the uncovered region is $\delta = (d+1)N_{arc}$.

The number of d-arcs in the border of the uncovered region, however, is data dependant. That is, depending on the data distribution and the value of ρ: the maximum distance between neighboring points. If data is distributed such that large clusters of data points are in each others neighborhood, most of such clusters are fully covered and most of the points in such region do not contribute to the complexity of the VC-dimension. Similarly, for the higher values of ρ, the chance of large regions being fully covered is higher and this makes the border of the region less complex. Also, if ρ is very small, almost all points belong to the uncovered region and, again, the uncovered region will be less complex.

Finding the number of d-arcs in the border of the uncovered region is challenging. On the other hand, the theoretical upper-bound in Theorem 3 required large numbers of samples and, depending on n and d, might not be practical. Therefore, in the following

Algorithm 2 UNCOVERED-MD

Input: dataset $\mathcal{D}$, k, ρ, ε, and ϕ

```
1: function IDENTIFY(D, k, ρ, ε, φ)
2:     T ← a set of i.i.d samples according to φ
3:     N_S ← an initial value // based on n, d
4:     repeat
5:         S ← N_s i.i.d samples from [0,1]^d
6:         for s ∈ S do
7:             if |{t ∈ D | Δ(t,s) ≤ ρ}| ≥ k then ℓ[s] ← +1
8:             else ℓ[s] ← -1
9:         U_a ← classifier(S, ℓ)
10:        Error← error(U_a, T); N_s ← 2 × N_s
11:    until Error≤ ε
12:    return (U_a, S, ℓ)
13: function REPORT(S, ℓ)
14:    T ← DecisitonTree(S, ℓ); desc ← {}
15:    for l ∈ T.leaves do
16:        if l.label= +1 then
17:            add (PathFromRoot(l), l.entropy) to desc
18:    return desc
19: function Query(q) return U_a(q)
```

we suggest an alternative approach for finding the right number of samples that satisfy the user-provided misclassification error ε. Specifically, we use an *exponential search* strategy: starting from an initial value for the number of samples, we iteratively double the number of samples until we find a proper number of samples.

4.1.3 Identifying the Uncovered Region. The function IDENTIFY for approximating the uncovered region when $d > 2$ is provided in Algorithm 2. The algorithm identifies the uncovered region in the form of a classifier $\mathcal{U}_a$ that, given a query point q, identifies if q belongs to the uncovered region. The algorithm uses $\mathcal{T}$, a set of i.i.d samples from the query space, as the test set to verify the misclassification error of $\mathcal{U}_a$. As we previously explained, we use iterative sampling and the exponential search strategy to identify the right number of samples. The algorithm starts by setting the sample set size N_s to an initial value. It then collects N_s i.i.d samples from the query space and labels them as +1 if those are covered by the dataset and -1 otherwise.

Recall from § 4.1 that we identify $\mathcal{U}_a$ as the (non-covex) hull around the -1 points that does not include *any* of the +1 (covered) points. In other words, viewing $\mathcal{S}$ as an ε'-**net**, a classifier with no mis-classification error on $\mathcal{S}$, is guaranteed to have the classification error of at most ε'. Algorithm 2 uses this observation and builds such classifier around $\mathcal{S}$.

After building the classifier, the algorithm checks if it satisfies the error requirement by the user; that is, if $\mathcal{U}_a$ has the miss-classification error (*Error*) of at most ε. To check this, the model uses the test set $\mathcal{T}$. If $Error > \varepsilon$, the algorithm doubles the sample size and repeats the process until it reaches the right sample size for N_s.

4.1.4 Complexity analysis of function IDENTIFY *of Algorithm 2.* The function uses exponential search in order to find the proper number of samples. Let $\mathcal{N}$ be the final number of samples. Since, starting from an initial value larger than 1, the algorithm doubles the sample size at each iteration, the total number of iterations is bounded by $O(\log \mathcal{N})$. At every iteration, the algorithm selects a sample set of size N_s and for each of them check if it is covered by the dataset. This, in worst-case, is in $O(n\,d)$ as for every tuple in the dataset, the algorithm requires to compute the distance between the sample point that the tuple. Of course, any of the off-the-shelf techniques and indices such as Local Sensitive Hashing can be used for efficient counting. After collecting the samples, the algorithm needs to train a classifier and compute its mis-classification error. Therefore, using T_c as the time required for training/testing the classifier for a sample set of size $\mathcal{N}$, the total time complexity for approximately identifying the uncovered region is $O\Big((\mathcal{N}\,n\,d + T_c)\log \mathcal{N}\Big)$.

4.2 Reporting the Uncovered Region in MD

The next step after identifying the uncovered region is to report it to the user. In 2D, we could create a map to visually present the region. This however is not possible, in particular when $d > 3$. While there are different choices to make, we believe while human being might not be very good at understanding complex geometric shapes, data scientists understand the axis-parallel range cubes[1]. For example, it seems easy to understand that the points in range $\langle x_1 : [.46, .58], x_2 : [.12, .18], x_3 : [.67, .81]\rangle$ are uncovered.

Ideally, we would like to identify a set $R = \{R_1, \cdots, R_\chi\}$ of axis-parallel hypercubes that best describe the uncovered region. That is, the number of uncovered points outside the hypercubes is minimized, and similarly the number of covered points within each hypercube is minimized. In other words, we would like to minimize $(\cup_{i=1}^{\chi} R_i) \oplus \mathcal{U}$. It turns out this problem is not "easy" to solve and exactly solving it is out of the scope of this paper.

Instead, as an approximation heuristic, we draw the connection to *decision trees*, as those are widely used for explaining black-box and complex AI models [25]. A decision tree is tree-structure classifier that expresses the target variable as a function of the values of other attributes. Every branch from a node in the tree is a "rule" based on an attribute which can be viewed as an axis-parallel hyperplane that splits the space in two halves. Following the cuts enforced by each of the branches, every leaf node in the tree describe an axis-parallel hypercube identified by the branches in the path from the leaf to the root of the tree. The arrangement of decision tree is such that (almost) all tuples in a leaf node have the same class label. A well-known approach for finding the tree partitions the space at every node greedily, based on a measure of entropy [53]. Following this argument, we build a decision tree using the samples and labels $(\mathcal{S}, \ell)$ drawn for identifying the uncovered region. As shown in the function REPORT in Algorithm 2, the corresponding ranges for +1 leaf nodes, together with their entropy, are reported as the description of uncovered regions to the user.

4.3 Query Answering

As specified in the function QUERY of Algorithm 2, the classifier $\mathcal{U}_a$ is being used to identify if a query point q is covered by the dataset $\mathcal{D}$. It approximates the uncovered region $\mathcal{U}$ and returns -1 if the queried point is uncovered.

[1]We recognize that other alternatives, such as reporting the center and the radius of hyper spheres covering the uncovered region could also be used here.

5 TUNING THE PARAMETERS ρ AND k

Similar to many other concepts such as clustering and frequent item-sets mining, the notion of coverage requires parameter tuning. In order to identify if a point is uncovered, we need to set the vicinity value ρ and the threshold k in Definition 1 (coverage). The techniques proposed in this paper are agnostic to the choice of these parameters. Nevertheless, in this section we present some heuristics for setting these parameters, prior to using these techniques.

The first parameter to determine is k: the minimum number of points in a neighborhood for a query point to be considered as covered. That is, the k points in the neighborhood of the point are the minimum representative to enable meaningful analysis about it. Following statistics and central limit theorem, the rule of thumb suggests the number of representatives should be around 30. For example, [60] suggests that a minimum of 20 to 50 samples is necessary.

The second parameter is the radius ρ, which defines the size of neighborhood considered. In the following, we propose two approaches for determining the value of ρ.

5.1 Absolute Choice Principle

Our first strategy is an absolute choice principle in which the parameters are chosen independently from how the data is distributed. The goal is to determine ρ: the maximum distance between two points to be considered in the neighborhood of each other. The smaller the parameter ρ is the more similar the neighboring points are. The value of ρ should be specified in a way that the tuples in the vicinity of a query point are similar enough to have similar "behaviors" with it. If so, the tuples in the ρ-neighborhood can be considered as the observation subgroup for learning and analyzing the behavior of the query point. Of course, the domain knowledge may be necessary for identifying the adjacency radius.

As a heuristic to identify the radius ρ, we consider the expected number of tuples in a neighborhood if the tuples were uniformly distributed. Let n be the number of tuples in $\mathcal{D}$. Knowing that all features have been normalized in range $[0, 1]$, the expected volume to contain k points is k/n. Using this argument, for a small constant $c \geq 1$ (e.g. $c = 2$), we can say if the volume of the neighborhood is c times larger than k/n, it should contain at least k tuples in it. The ρ-neighborhood of a point q in the d dimensional space is a d dimensional hypersphere with radius ρ. The volume of such a d-sphere [44] is

$$V_d(\rho) = \frac{\pi^{d/2}}{\Gamma(d/2+1)}\rho^d \tag{2}$$

Specifically, for $d = 2$ and $d = 3$ the volumes are $V_2(\rho) = \pi\rho^2$ and $V_3(\rho) = 4/3\pi\rho^3$, respectively. Setting $V_d(\rho)$ as ck/n, ρ is computed as:

$$\rho = \frac{1}{\sqrt{\pi}}\left(\frac{c\,k\,\Gamma(d/2+1)}{n}\right)^{\frac{1}{d}} \tag{3}$$

5.2 Sensitive Region Analysis

Our next heuristic for determining ρ follows a relative choice principle. That is, rather than finding the uncovered region, we aim to find the *"most uncovered"* region.

For a small value of ρ a large portion of the value space may be uncovered. As the value of ρ increases the volume of the uncovered region decreases. Specifically, moving from $\rho = 0$ to $\rho = 1$ the volume of the uncovered region decreases from one to zero. Apparently, the points that still remain uncovered while increasing ρ are more problematic as the tuples in the dataset are not similar to those. In addition, at least for the purpose of reporting dataset deficiencies to the data scientist, reporting all (or none of the) possible points as uncovered is not informative. Ideally, a data scientist may want to see a small-enough ratio of value space as the most uncovered (most problematic) region. This is similar to clustering where putting all points in one cluster (or each point in a separate cluster) may not seem beneficial; a "good" clustering is supposed to return a few clusters.

Following the above argument, given a value $0 < \alpha < 1$ (e.g. $\alpha = 0.1$), our objective to find a value of ρ such that a portion α of the value space (e.g. ten percent) is uncovered. In order to find the value of ρ, we pay attention to the fact that α monotonically decreases as ρ increases. This allows us to apply binary search for finding the corresponding value ρ for α. Let T be the time required to identify the uncovered region for the given values of ρ and k. The binary search calls the uncovered region identification function $\log \frac{1}{\epsilon}$ times and hence it runs in $O(T \log \frac{1}{\epsilon})$.

6 EXPERIMENTS

We conducted comprehensive experiments on real datasets both to validate our proposal and to evaluate the performance of our algorithms.

6.1 Experiments Setup

The experiments were conducted using a 2.5 GHz Quad-Core Intel Core i7 processor, 16 GB memory, running macOS. The algorithms were implemented in Java and Python.

We used four real datasets for our experiments.

- *3D Road Network (RN) Dataset [39]* is a benchmark dataset for regression that was constructed by adding elevation information to a 2D road network in North Jutland, Denmark. It includes 434,874 records with attributes `Latitude`, `Longitude`, and `Altitude`.
- *City of Chicago's Food Inspections (FI) Dataset [51]* contains inspections of restaurants and other food establishments in Chicago from Jan. 2010 to June 2018. The dataset contains 210,268 tuples and 17 features, including `Latitude` and `Longitude`.
- *NBA dataset [2]* contains the points for each combination of player/team/season up to 2009. It contains 21,961 tuples and the ordinal attributes `gp`, `minutes`, `pts`, `oreb`, `dreb`, etc.
- *US Department of Transportation flights database (DOT) [3]* is widely used for identifying on-time flight performance. After removing records with missing values, the dataset contains 457,892 records, for all flights conducted in the last months of 2017, over 8 scalar attributes, including `Dep-Delay`, `Taxi-Out`, and `Air-time`.

All values used for studying coverage are normalized in the range $[0, 1]$, using $(v_i - \min)/(\max - \min)$.

Evaluation plan and performance measures: We evaluate the performance of our algorithms UNCOVERED-2D and UNCOVERED-MD in this section. Time is our main metric. Unlike UNCOVERED-2D,

Figure 2: Cat Image, Illustration of classifier's failure on FN rate for uncovered region

UNCOVERED-MD is an approximation algorithm whose performance depends on the number of samples required for building the ϵ-**net**. Here. we report the number of samples UNCOVERED-MD takes, using its exponential search strategy. In addition, we also report the coverage ratio, i.e., the fraction of the query space occupied by covered regions, where every point is covered by the dataset.

Default values: To evaluate the performance of our algorithms under different settings, we vary the value of a parameter, while fixing the value of other ones. The default value for k is 30 and ρ is set to 0.1 for 2D and 0.15 for MD. The default misclassification error is $e = 0.01$. That is, we aim to approximate the uncovered region $\mathcal{U}$ with $\mathcal{U}_a$ such that $e = |\mathcal{U} \oplus \mathcal{U}_a| \leq 0.01$. The default value for d (number of attributes) is 3. Default value of n for the FI, NBA, and DoT datasets is 10K, 100K, and 20K, respectively.

6.2 Proof of Concept

We begin our experiments with a demonstration of why coverage matters. We select a classification and a regression task that are both easy to visualize. In both cases, we show that (a) lack of coverage can cause low performance of the model, and (b) resolving lack of coverage resolves the poor performance of the model for uncovered region.

6.2.1 Classification. Consider a classification task to label a query point on the x-y plane, as belonging to the body of a cat image or to the background. We selected the image shown in Figure 2 with a resolution of 4106×2720. We generated the training data of size 11K by randomly sampling from the image with a sampling ratio of 0.001 and labeling each sample point as +1 if inside the cat body, -1 otherwise. Next, we intentionally removed the sample points in the training data that belong to the patch highlighted in the figure to make it uncovered. Following the same procedure, we generated 30 datasets and repeated each experiment 30 times, using different datasets. Using this as the training data, we tried different classification models, namely, Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), kNN, and SVM. The linear classifiers LR and SVM did not perform well. DT had the highest overall performance compared to other classifiers and all classifiers failed to work for the uncovered region. On average, the overall F1 measure for the DT classifier was 95% but it was 62% for the uncovered region. In particular, while the overall FN rate was less than 5%, it was as high as 54% for the uncovered region. To further investigate this, in Figure 2, we painted the parts of the patch that model mistakenly labeled as -1 in red. The figure is self-explanatory. Relying on its training data, it created the decision boundary by connecting the two edges of the cat body, missing its ear. As a result, the query points that belong to the ear are misclassified as background, resulting in the high FN rate. Next, in order to confirm the issue was due to the lack of coverage, we gradually added 1%, 10% and 100% of sample points from the uncovered region back to the training data. The results are provide in Figure 3. Looking at the figure, adding sample to the patch increased the coverage ratio in the patch from 0.6 to 1. At the same time the F1 and FN rate measures converged to the same numbers as other regions. In particular, after resolving the lack of coverage, FN rate for the patch moved from 39% down to less than 2%.

6.2.2 Regression. As our next experiment, we considered a regression task, using the RN dataset. Given a query point in form of (long, lat), the objective is to predict the altitude. We build the model, randomly sampling the dataset with a ratio of 10%, we generated 30 training datasets of size 43K. Similar to the previous example, we removed the samples from a cell in the range 10<longitude<10.6 and 57.1<latitude<57.6 to make it uncovered. We intentionally selected this cell since the altitude fluctuations was high in that cell. Using the training data, we tried three different predictive models, namely (1) ElasticNet, (2) DT, and (3) kNN. DT outperformed the other models while all three models had high prediction error for the uncovered region. While the mean squared error was close to zero, it was 714.8 for the uncovered region. Next, in Figure 4 we gradually added sample points from the uncovered patch to the training data. Upon adding the sample points, the coverage ratio increased from 2% up to 92% and, hence, MSE dropped down to 29.

Before concluding this section, we would like to reiterate lack of coverage does not necessarily mean that the model will perform poorly for those regions. For example, in our classification experiment if the uncovered region were fully inside (or outside) the cat body, the classifier could have performed well even for the uncovered region. However, one cannot in general know whether we are in a problematic "ear-like" region or not, without additional information beyond the available training data. Furthermore, whether this matters is also task-specific. So, at the least, an uncovered query point should come with a warning that users can choose to heed or ignore based on other knowledge available to them.

6.3 Performance Evaluation

In the following, we provide our experiment results, using the aforementioned settings. First in § 6.3.1, we evaluate a baseline adapted from the existing work [13]. Next in § 6.3.2, we evaluate the UNCOVERED-2D algorithm, using FI and RN datasets. Finally, in § 6.3.3, we conduct the performance evaluation for UNCOVERED-MD using the NBA and DoT datasets.

6.3.1 Baseline. Before evaluating our algorithms, we first consider adapting the existing [13] for identifying uncovered region in (low-cardinality) discrete attributes. Since the space is continuous here, we first discretize it using grids of different granularity. Then, we

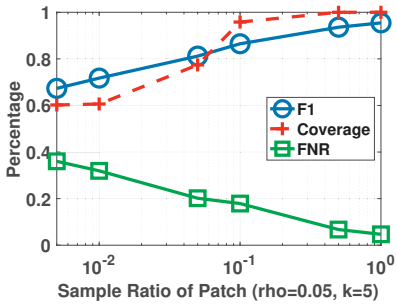


Figure 3: Cat Image, impact of lack of coverage for Classification

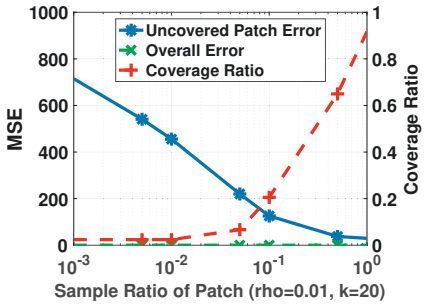


Figure 4: RN, impact of lack of coverage for Regression

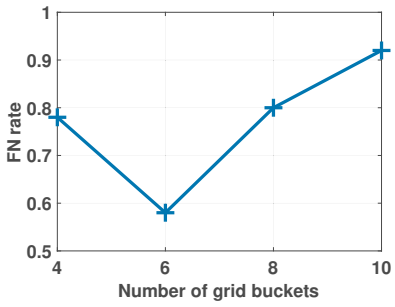


Figure 5: RN, DeepDiver's error in detecting Uncovered points

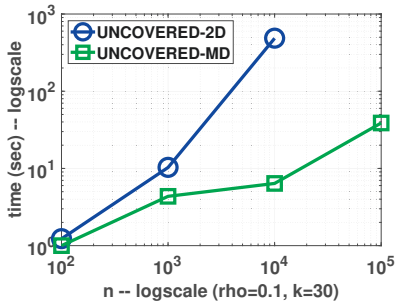


Figure 6: FI, impact of n on time

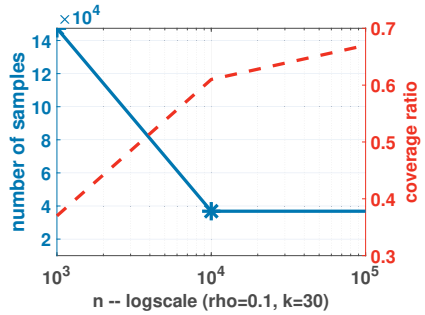


Figure 7: FI, impact of n on sample size and coverage

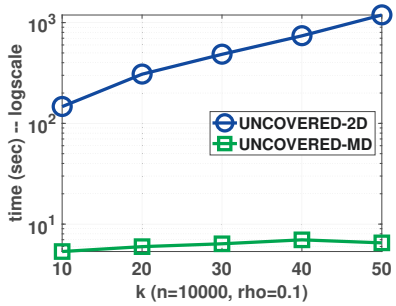


Figure 8: FI, impact of k on time

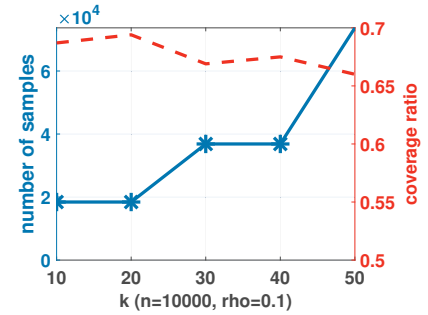


Figure 9: FI, impact of k on sample size and coverage

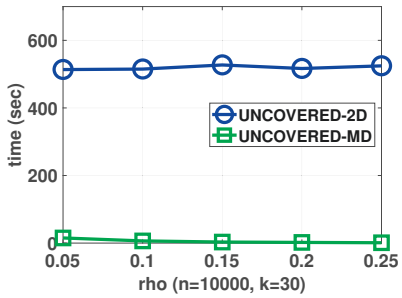


Figure 10: FI, impact of ρ on time

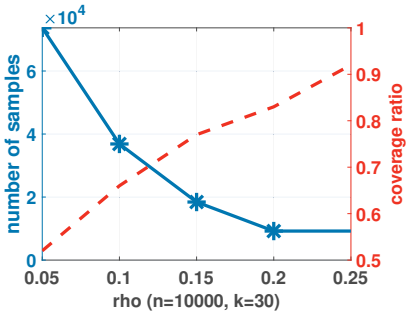


Figure 11: FI, impact of ρ on sample size and coverage

use the algorithm DeepDiver in order to discover the uncovered region in form of maximal uncovered patterns (MUPs). The union of the space covered by the MUPs is considered as the uncovered region. DeepDiver requires a parameter k for defining coverage. Since it is not clear how to transfer (ρ, k) to k in discrete space, to have a fair comparison, for every setting we first discover the uncovered region using our algorithms in this paper and then apply a binary search on the value of k, to find the setting that best matches the continuous uncovered region.

Using a random subset of size 900 from RN, we used this algorithm on different grid granularities for $d = 3$ in order to find the uncovered region for $\rho = 0.02$ and $k = 10$. We then used a test set of 100 query point randomly sampled from RN to evaluate the performance of the algorithm. In order to find if a query point is uncovered, we need to traverse among the MUPs to see if it matches any of those patterns. This can become inefficient for fine-grained where the number of MUPs can be as high as the number of buckets to the power of d. Efficiency aside, this baseline approach failed to discover a large portion of uncovered points, as shown in Figure 5. Looking at the figure, in all cases, the algorithm mistakenly labeled more than 58% of uncovered points as covered. Also, even though increasing the number of buckets initially helped to reduce the

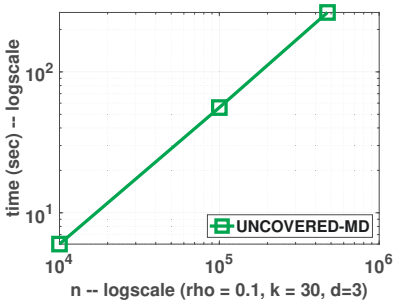


Figure 12: DOT, impact of n on time

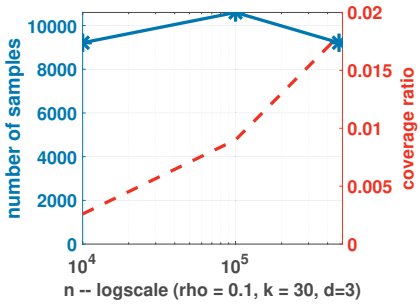


Figure 13: DOT, impact of n on sample size and coverage

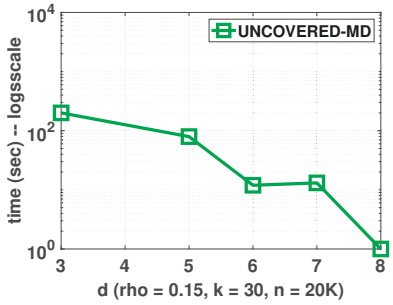


Figure 14: NBA, impact of d on time

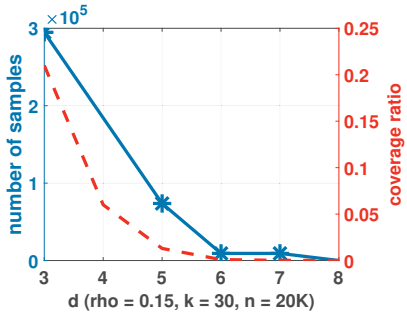


Figure 15: NBA, impact of d on sample size and coverage

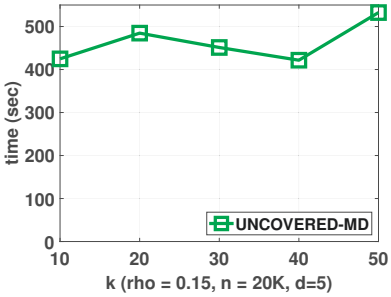


Figure 16: NBA, impact of k on time

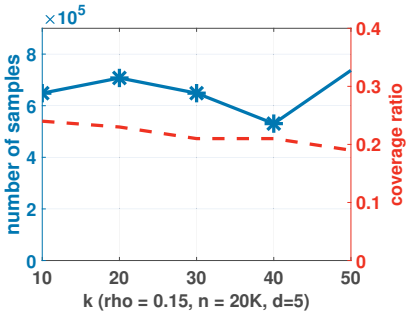


Figure 17: NBA, impact of k on sample size and coverage

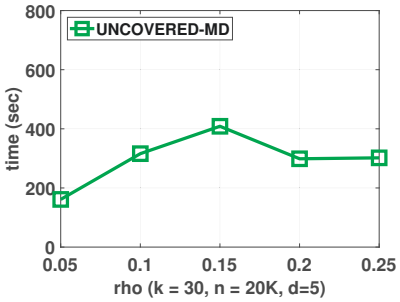


Figure 18: NBA, impact of ρ on time

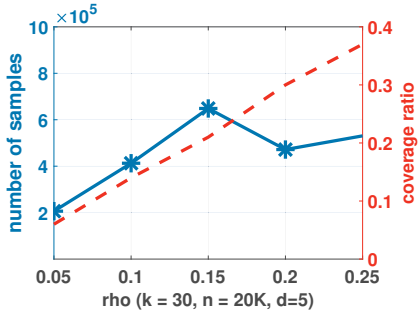


Figure 19: NBA, impact of ρ on sample size and coverage

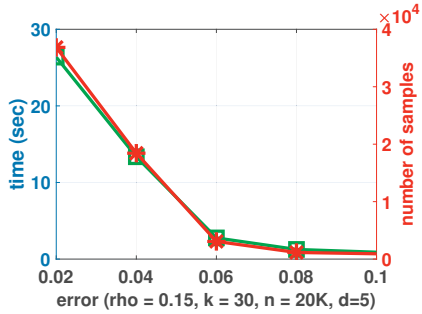


Figure 20: NBA, impact of error e on time and sample size

error, after 6 bucket the error increased again up to 92%. The reason is that as the cells get very small, the corresponding number of samples they should contain for coverage reduces. As k decreases, the algorithm has increased error in detecting uncovered regions, particularly near the borders.

6.3.2 2D Experiments. In this section, we use the RN and FI datasets (using `Latitude` and `Longitude` as the attributes) to evaluate the UNCOVERED-2D algorithm. We obtained similar results with almost identical plots for the two datasets. Therefore, in the following we present the results for one of the datasets (FI) with different settings.

Varying n: In this experiment, we study the impact of the number of items in the performance of the algorithm and in the coverage. To do so, we change n from 100 to 100K. We then use our exact UNCOVERED-2D algorithm and the sampling-based algorithm UNCOVERED-MD for identifying the uncovered region. The results are provided in Figures 6 and 7. As expressed in Lemma 2, using the advanced algorithms such as [42] for finding the k-th order voronoi diagram, UNCOVERED-2D is in $O(k^2 n \log n)$. We used off-the-shelf implementations for constructing the k-th order voronoi diagram which scaled up to $n = 10K$ within a few minutes. The

UNCOVERED-MD algorithm, on the other hand, efficiently approximated the uncovered region with 99% accuracy in less than a minute in all settings. Increasing the dataset sizes increases the chance that a given query has enough items in its neighborhood to become covered. This is confirmed in Figure 7. The left-y-axis in this plot (and other similar plots in this section) shows the number of samples UNCOVERED-MD used for identifying the uncovered region and the right-y-axis (and the dashed orange line) show the coverage ratio. From the figure, increasing n from 1K to 100K, the coverage ratio increases up to 0.7. The increase in the coverage ratio makes the border of the uncovered region less complex, resulting in less number of samples required. Note that this is in contrary with our adversarial upper-bound in Theorem 3. The reason is that even though increasing n increases the number of voronoi cells, but most of them become completely covered, resulting in a less complex uncovered region.

Varying k and ρ: In our next experiments, we study the impact of varying the coverage parameters, i.e., the vicinity value ρ and the count threshold k. Figures 8 and 9 show the results for varying k (while fixing the other parameters to their default values). First, increasing the value of k increases the number of voronoi cells and, hence, the time taken by the UNCOVERED-2D algorithm. The performance of UNCOVERED-MD, on the other hand, depends on the number of samples required by the algorithm. The increase in k reduces the chance of a query point to be covered by the dataset, hence reducing the ratio of covered region to 65%. As a result, more voronoi cells become partially uncovered, increasing the complexity of the border of the uncovered region, resulting in the need for more number of samples. The other parameter that impacts the uncovered region is ρ. The results for varying this parameter are provided in Figures10 and 11. The value of ρ does not impact the number of voronoi cells, hence not impacting the UNCOVERED-2D runtime. The vicinity radius, however, does impact the coverage ratio and the complexity of the uncovered region, which impacts the number of samples required by UNCOVERED-MD. Increase the value of ρ increases the chance of an arbitrary query point to become covered by the dataset, hence increases the coverage ratio. As more and more voronoi cells become completely covered, the uncovered region becomes less complex and requires less number of samples to identify it.

6.3.3 MD experiments. We use the NBA dataset and DoT dataset for studying the performance of UNCOVERED-MD, when $d > 2$. In particular, we use the DoT dataset for varying the number of items up to 0.5 million, and the NBA dataset for the other settings. In all the settings, we aim to satisfy the mis-classification error of 1%.

Varying n: In this experiment, we use the DoT dataset to study the impact of vary the number of items up to half a million. The experiment results are provided in Figure 12 and 13. Similar to our Food Inspection experiment, the UNCOVERED-MD algorithm could scale to the large settings, finishing in a few minutes for $n = 0.5M$. We also observed a similar performance for the NBA dataset. Besides evaluating the performance of the algorithm, this experiment highlights the major coverage issue of the DoT dataset. Looking at the right-y-axis of Figure 13, one can observe even in a dataset with half a million items, more than 98% of the query space is uncovered. This

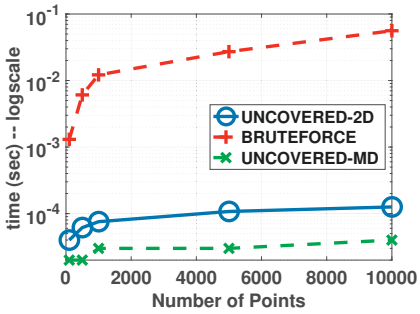


Figure 21: RN, Query time comparison.

is because, all items have very similar values. Note that the large portion of the uncovered region does not mean that a large ratio of query points are going to be uncovered. Assuming that training data and the query points follow the underlying distribution, most of the query points should be covered. We verified this for a test set of 200 random samples and all of them were covered, even though 98% of the space was uncovered. The values are even closer to zero for the attributes such as `departure-delay` and `arrival-delay`. As a result, while almost all the dataset records are condensed around the origin, most value combinations (possible query points) had less than 30 items in their neighborhoods, resulting in the significantly large uncovered region. Note that, increasing the number of attributes, the coverage ratio gets further decreased. We also observed that decreasing the values of k or increasing the value of ρ did not significantly change the coverage ratio. Therefore, for the rest of the experiments we use the NBA dataset.

Varying d, k, ρ, and the error threshold: In this experiment, we use the NBA dataset and, setting the other parameters to their default values, vary the number of attributes d, the coverage parameters k and ρ, and the error threshold for the algorithm. Figures 14 and 15 show the results for varying d from 3 to 8. Increasing the number of dimensions increases the distance between the points in the space, resulting in a quick decrease in the coverage ratio. As the coverage ratio decreases, more and more voronoi cells become completely uncovered. Hence the uncovered region becomes less complex, requiring less number of samples for identifying it which results in the decrease in the running time of UNCOVERED-MD. The results for varying k (resp. ρ) are provided in Figures 16 and 17 (resp. Figures 18 and 19). Increasing the value of k did not significantly reduce the coverage ratio, and therefore, the number of samples and the running time of UNCOVERED-MD did not significantly change. The increase in the value of ρ, on the other hand, increased the coverage ratio from less than 10% to around 40%. The increase in the coverage ratio initially increased the complexity of the uncovered region, causing an increase in the running time of UNCOVERED-MD. After the coverage ratio of 25%, the increase in the ratio, did not add to the complexity of the uncovered region, not meaningfully changing the running time. Finally, Figure 20 shows that, as expected, the approximation error threshold has a major impact on the running time of UNCOVERED-MD and the number of samples required for approximately identifying the uncovered region.

Query time comparison: After evaluating the performance of the UNCOVERED-2D and UNCOVERED-MD algorithms in identifying uncovered regions, we evaluate the query time of the two algorithms, using the RN dataset and 100 random queries for each setting. Additionally, we considered the brute-force approach that (assuming the availability of the training data at run-time), traversing through the training data, counts the number of neighbors of a point and stops as soon as it realizes the point is covered. Figure 21 shows the query time for different values of n. Having a logarithmic run-time, UNCOVERED-2D significantly outperformed the linear brute-force algorithm in all cases. In all cases UNCOVERED-MD was even faster than UNCOVERED-2D. Both algorithms were fast enough to be considered for online query answering.

7 RELATED WORK

The general topic of responsible data science has recently become timely across different research communities and the conference ACM FAccT (previously FAT*) has been dedicated to this topic. In particular, in past few years the database community has made several advancement to this topic [8, 10–13, 32, 33, 37, 38, 41, 45, 54–56, 59, 61, 63, 64].

Bias in data have been looked at for a long time in statistical community [49] but social data presents different challenges [17, 18, 26, 40, 50]. The diversity and representativeness of data have been widely studied [27], in fields such as social science [19, 24, 57], political science [62], and information retrieval [5]. Related work also includes [46–48] that aim to identify the largest empty hyper-rectangles in a dataset. To do so, they sequentially add data points to the region and cut the empty regions containing them in two smaller ones. The notion of coverage over low-dimensional categorical attributes has been proposed in [13, 38, 45]. In particular, uncovered regions are identified in form of value combinations (e.g. Hispanic Females) called patterns. A pattern is uncovered if there are less than k samples matching it. [22] analyzes the training set coverage of the protected attribute contributing to machine learning discrimination. Still, to the best of our knowledge, our paper is the first to extend the notion of coverage to continuous-valued attributes.

The techniques proposed in this paper heavily rely on the extensive research and advanced algorithms for voronoi diagrams [14–16, 20, 29–31, 42] and approximation geometric algorithms [34].

8 EXTENSIONS AND FUTURE WORK

Combination of continuous and discrete attributes: In this paper, we so far assumed all attributes ordinal and normalized in the range [0, 1]. Extending our notion of coverage to a mix of ordinal and non-ordinal attributes requires defining a proper notion of distance in a way that "neighborhood" becomes meaningful. We rely on existing work for such extension. In particular, [6, 36] defined a distance measure by combining the square Euclidean distance for numeric variables and simple matching distance for categorical variables. Similarly, a generalized distance measure to combine numeric, categorical, and binary attributes has been proposed in [35]. After defining the distance function, Definitions 1 and 2, as well as the MD algorithms in § 4, naturally extend. The extension of the 2D algorithm, however, depends on the complexity and the existence proper k-Voronoi diagrams for the new distance measure. Further investigation, in future work, may find more improvements.

High dimensional coverage: As reflected in Theorem 3, due to the curse of dimensionality, the MD algorithm proposed in this paper may not scale to very high dimensions. Still, we note that it is popular for such cases to first apply dimension reduction techniques such as PCA. We leave algorithm design for high dimensional cases for future work.

Coverage improvement: In this paper we proposed the notion of coverage over ordinal attributes and proposed algorithms to identify the uncovered region. An interesting question for future work is how to effectively mitigate this lack of coverage when acquiring labeled data is expensive.

9 CONCLUSION

Good data preparation is central to getting good results from AI and data science. Ensuring adequate coverage of feature space in the training data is one important aspect of good data preparation. Coverage has previously been studied only in discrete categorical feature spaces. In this paper, we showed how to think about coverage in continuous space, and developed efficient algorithms both to identify poorly covered regions of space at model learning time and to issue warnings of poor coverage when appropriate at query time. We believe addressing such problems is central to data science and AI, and are among the many contributions data management experts can make to these fields.

10 ACKNOWLEDGEMENTS

Abolfazl Asudeh was supported in part by Google Research Scholar Award. H. V. Jagadish was supported in part by NSF grants No. 1741022 and 1934565.

REFERENCES

[1] [n. d.]. COMPAS Recidivism Risk Score Data and Analysis. www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis.
[2] [n. d.]. NBA players statistics. www.databasebasketball.com/. Accessed: 2016.
[3] [n. d.]. US Department of Transportation. www.transtats.bts.gov/DL_SelectFields.asp?. Accessed: 2018.
[4] Pankaj K Agarwal, Mark De Berg, Jirí Matousek, and Otfried Schwarzkopf. 1998. Constructing levels in arrangements and higher order Voronoi diagrams. *SIAM journal on computing* 27, 3 (1998), 654–667.
[5] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *WSDM*. ACM, 5–14.
[6] Amir Ahmad and Lipika Dey. 2007. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering* 63, 2 (2007), 503–527.
[7] Matthew Arnold, Rachel KE Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, A Mojsilović, Ravi Nair, K Natesan Ramamurthy, Alexandra Olteanu, David Piorkowski, et al. 2019. FactSheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM Journal of Research and Development* 63, 4/5 (2019), 6–1.
[8] Abolfazl Asudeh. 2021. Enabling Responsible Data Science in Practice. *ACM SIGMOD Blog* (January 2021).
[9] Abolfazl Asudeh, HV Jagadish, Gerome Miklau, and Julia Stoyanovich. 2018. On obtaining stable rankings. *Proceedings of the VLDB Endowment* 12, 3 (2018), 237–250.
[10] Abolfazl Asudeh, HV Jagadish, Gerome Miklau, and Julia Stoyanovich. 2019. On Obtaining Stable Rankings. *PVLDB* 12, 3 (2019).
[11] Abolfazl Asudeh, HV Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing fair ranking schemes. In *SIGMOD*. 1259–1276.
[12] Abolfazl Asudeh and H. V. Jagadish. 2020. Fairly evaluating and scoring items in a data set. *PVLDB* 13, 12 (2020), 3445–3448.

[13] Abolfazl Asudeh, Zhongjun Jin, and HV Jagadish. 2019. Assessing and remedying coverage for a given dataset. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 554–565.

[14] Franz Aurenhammer. 1991. Voronoi diagrams-a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)* 23, 3 (1991), 345–405.

[15] Franz Aurenhammer and Rolf Klein. 2000. Voronoi diagrams. *Handbook of computational geometry* 5, 10 (2000), 201–290.

[16] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. 2013. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company.

[17] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2019. Fairness and machine learning: Limitations and opportunities. fairmlbook.org.

[18] Solon Barocas and Andrew D Selbst. 2016. Big data's disparate impact. *Calif. L. Rev.* 104 (2016), 671.

[19] Ellen Berrey. 2015. *The enigma of diversity: The language of race and the limits of racial justice*. University of Chicago Press.

[20] Cecilia Bohler, Panagiotis Cheilaris, Rolf Klein, Chih-Hung Liu, Evanthia Papadopoulou, and Maksym Zavershynskyi. 2013. On the complexity of higher order abstract Voronoi diagrams. In *International Colloquium on Automata, Languages, and Programming*. Springer, 208–219.

[21] Bernard Chazelle and Herbert Edelsbrunner. 1987. An improved algorithm for constructing kth-order Voronoi diagrams. *IEEE Trans. Comput.* 100, 11 (1987), 1349–1354.

[22] Irene Chen, Fredrik D Johansson, and David Sontag. 2018. Why is my classifier discriminatory?. In *NeurIPS*. 3539–3550.

[23] Jeffrey Dastin. 2018. Amazon scraps secret AI recruiting tool that showed bias against women. *San Fransico, CA: Reuters. Retrieved on October* 9 (2018), 2018.

[24] Frank Dobbin and Alexandra Kalev. 2016. Why diversity programs fail and what works better. *Harvard Business Review* 94, 7-8 (2016), 52–60.

[25] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. 2018. Explainable artificial intelligence: A survey. In *MIPRO*. IEEE, 0210–0215.

[26] Marina Drosou, HV Jagadish, Evaggelia Pitoura, and Julia Stoyanovich. 2017. Diversity in big data: A review. *Big data* 5, 2 (2017), 73–84.

[27] Marina Drosou, HV Jagadish, Evaggelia Pitoura, and Julia Stoyanovich. 2017. Diversity in big data: A review. *Big data* 5, 2 (2017).

[28] Herbert Edelsbrunner, Nany Hasan, Raimund Seidel, and Xiao Jun Shen. 1989. Circles through two points that always enclose many points. *Geometriae Dedicata* 32, 1 (1989), 1–12.

[29] Herbert Edelsbrunner and Raimund Seidel. 1986. Voronoi diagrams and arrangements. *Discrete & Computational Geometry* 1, 1 (1986), 25–44.

[30] Steven Fortune. 1987. A sweepline algorithm for Voronoi diagrams. *Algorithmica* 2, 1-4 (1987), 153.

[31] Steven Fortune. 1995. Voronoi diagrams and Delaunay triangulations. In *Computing in Euclidean geometry*. World Scientific, 225–265.

[32] Lise Getoor. 2019. Responsible Data Science. In *SIGMOD*.

[33] Yifan Guan, Abolfazl Asudeh, Pranav Mayuram, HV Jagadish, Julia Stoyanovich, Gerome Miklau, and Gautam Das. 2019. Mithraranking: A system for responsible ranking design. In *SIGMOD*. 1913–1916.

[34] Sariel Har-Peled. 2011. *Geometric approximation algorithms*. Number 173. American Mathematical Soc.

[35] Sandhya Harikumar and PV Surya. 2015. K-medoid clustering for heterogeneous datasets. *Procedia Computer Science* 70 (2015), 226–237.

[36] Zhexue Huang. 1997. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)*. Citeseer, 21–34.

[37] HV Jagadish, Francesco Bonchi, Tina Eliassi-Rad, Lise Getoor, Krishna Gummadi, and Julia Stoyanovich. 2019. The Responsibility Challenge for Data. In *SIGMOD*. 412–414.

[38] Zhongjun Jin, Mengjing Xu, Chenkai Sun, Abolfazl Asudeh, and HV Jagadish. 2020. MithraCoverage: A System for Investigating Population Bias for Intersectional Fairness. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2721–2724.

[39] Manohar Kaul, Bin Yang, and Christian S Jensen. 2013. Building accurate 3d spatial networks to enable next generation intelligent transportation systems. In *2013 IEEE 14th International Conference on Mobile Data Management*, Vol. 1. IEEE, 137–146.

[40] Jon Kleinberg. 2019. Fairness, Rankings, and Behavioral Biases. FAT*.

[41] Caitlin Kuhlman and Elke Rundensteiner. 2020. Rank aggregation algorithms for fair consensus. *PVLDB* 13, 12 (2020), 2706–2719.

[42] Der-Tsai Lee. 1982. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE transactions on computers* 100, 6 (1982), 478–487.

[43] Jerry Zheng Li. 2018. *Principled approaches to robust machine learning and beyond*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[44] Shengqiao Li. 2011. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics* 4, 1 (2011), 66–70.

[45] Yin Lin, Yifan Guan, Abolfazl Asudeh, and HV Jagadish. 2020. Identifying insufficient data coverage in databases with multiple relations. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2229–2242.

[46] Bing Liu, Wynne Hsu, and Shu Chen. 1997. Using General Impressions to Analyze Discovered Classification Rules. In *KDD*. 31–36.

[47] Bing Liu, Liang-Ping Ku, and Wynne Hsu. 1997. Discovering interesting holes in data. In *IJCAI*. Springer, 930–935.

[48] Elsa Loekito and James Bailey. 2006. Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams. In *SIGKDD*. ACM, 307–316.

[49] Jerzy Neyman and Egon Sharpe Pearson. 1936. Contributions to the theory of testing statistical hypotheses. *Statistical Research Memoirs* (1936).

[50] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Emre Kiciman. 2019. Social data: Biases, methodological pitfalls, and ethical boundaries. *Frontiers in Big Data* 2 (2019), 13.

[51] Chicago Data Portal. [n. d.]. Food Inspections Dataset. https://data.cityofchicago.org/Health-Human-Services/Food-Inspections-1-1-2010-6-30-2018/puke-h9vk. Accessed: 2016.

[52] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[53] Lior Rokach and Oded Maimon. 2005. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35, 4 (2005), 476–487.

[54] Babak Salimi, Bill Howe, and Dan Suciu. 2020. Database Repair Meets Algorithmic Fairness. *ACM SIGMOD Record* 49, 1 (2020), 34–41.

[55] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional fairness: Causal database repair for algorithmic fairness. In *SIGMOD*. 793–810.

[56] Nihar B Shah and Zachary Lipton. 2020. SIGMOD 2020 Tutorial on Fairness and Bias in Peer Review and Other Sociotechnical Intelligent Systems. In *SIGMOD*. 2637–2640.

[57] Edward H Simpson. 1949. Measurement of diversity. *Nature* 163, 4148 (1949).

[58] Julia Stoyanovich and Bill Howe. 2019. Nutritional Labels for Data and Models. *IEEE Data Eng. Bull.* 42, 3 (2019), 13–23.

[59] Julia Stoyanovich, Bill Howe, and HV Jagadish. 2020. Responsible data management. *PVLDB* 13, 12 (2020), 3474–3488.

[60] Seymour Sudman. 1976. Applied sampling. *Academic Press New York* (1976).

[61] Chenkai Sun, Abolfazl Asudeh, HV Jagadish, Bill Howe, and Julia Stoyanovich. 2019. Mithralabel: Flexible dataset nutritional labels for responsible data science. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2893–2896.

[62] James Surowiecki. 2005. *The wisdom of crowds*. Anchor.

[63] Suresh Venkatasubramanian. 2019. Algorithmic fairness: Measures, methods and representations. In *PODS*. 481–481.

[64] Ke Yang, Julia Stoyanovich, Abolfazl Asudeh, Bill Howe, HV Jagadish, and Gerome Miklau. 2018. A nutritional label for rankings. In *SIGMOD*. 1773–1776.