Patterns Count-Based Labels for Datasets

Yuval Moskovitch
University of Michigan
yuvalm@umich.edu

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

2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00184

***Abstract*—Counts of attribute-value combinations are central to the profiling of a data set, particularly in determining fitness for use and in eliminating bias and unfairness. While counts of individual attribute values may be stored in some data set profiles, there are too many combinations of attributes for it to be practical to store counts for each combination. In this paper, we develop the notion of storing a "label" of limited size that can be used to obtain good estimates for these counts. A label, in this paper, contains information regarding the count of selected attribute-value combinations (which we call "patterns") in the data. We define an estimation function, that uses this label to estimate the count of every pattern. We present the problem of finding the optimal label given a bound on its size and propose a heuristic algorithm for generating optimal labels. We experimentally show the accuracy of count estimates derived from the resulting labels and the efficiency of our algorithm.**

I. Introduction

Data-driven decision systems are increasingly being used in a wide range of domains, where data-driven algorithmic decision-making may affect human life. For instance, risk assessment tools, which predict the likelihood of a defendant to re-offend, are widely used in courtrooms across the US [4] to make decisions about bail. The data on which these systems depend, as in much of data science, are often "found data", namely, data that was not collected as part of the development of the analytics pipeline, but was acquired independently, possibly assembled by others for different purposes. When the decision is made by a machine-learned model, the correctness and quality of the decision depend centrally on the data used in the model training phase. In particular, the use of improper, unrepresentative, or biased data may lead to unfair decisions, algorithmic discrimination (such as racism), and biased models [8].

ProPublica, a non-profit newsroom that produces investigative journalism in the public interest, conducted a study on the risk assessment scores output by a widely used software program developed by Northpointe, Inc. They found that the program discriminated based on race: blacks were scored at greater risk of re-offending than the actual, while whites were scores at lower risk than actual. Further analysis [5] showed issues with other groups as well. For example, the error rate for Hispanic women is very high because there aren't many Hispanic women in the data set. It is not only that there are fewer Hispanics than blacks and whites, and fewer women then men, but also fewer Hispanic women than one would expect if these attribute values were independently distributed. A judge sentencing a Hispanic woman presumably would like to be informed about this low count of Hispanic women in the data set and the consequent likelihood of greater error in the risk assessment.

Information regarding the attributes' values such as their type, distribution statistics, common patterns, and attributes correlations and dependencies may assist in mitigating misuse of data and reduce algorithmic bias and racism. This flavor of information can be extracted in the process of data profiling, a standard step performed by analysts when using "found data". While informative and useful, data profiling is hard to do well, is usually not automated, and requires significant effort. To help both the data analyst and the data user, the notion of a "nutrition label" has been suggested [7], [9], [10], [14]–[16]. The basic idea of a nutrition label is to capture, in a succinct label, data set properties of interest. Perhaps the single most important such property is a profile of the counts of various attribute value combinations. For instance, an analyst may wish to ensure a (close) to real-world distribution in the attribute's values of the data, such as an equal number of males and females. Another concern may be the lack of adequate representation in the data for a particular group [5], such as divorced African-American females, or contrarily, too high a percentage of data that represents the same group (data skew) [6]. The count information may also reveal potential dependent or correlated attributes. As a simple example, if all tuples representing individuals under 20 years old are also single, this may point out a possible connection between age and marital status.

In this paper, we propose to label datasets with information regarding the count of different patterns (attributes values combinations) in the data. Storing individual counts for each possible combination is likely to be impossible as their number is exponential in the number of attributes. To this end, we focus on techniques to estimate these counts using only a limited amount of information. Given a dataset, if we do not know anything about value distributions in it, a common assumption to make is that of independence between attributes. Under this assumption, count estimation of attribute-value combinations can be done using only the counts of individual attribute values. However, this defeats the central purpose of profiling – we only get information about individual attributes (the "marginal distributions") but nothing about any correlations. In the study of discrimination, there is a considerable examination of *intersectionality*, the whole point of which is to understand

This research has been supported in part by NSF grants 1741022 and 1934565.

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00184

how the social consequence of being a member of a protected class on multiple axes is not simply the "sum" of each alone. For example, to understand the discrimination faced by black women it is not enough to understand independently the impact of race alone and gender alone. In other words, we have to ensure that our estimates for the count of any pattern in the database are at least approximately correct.

Our problem, intuitively, is to choose a small number of patterns (limited by a given space budget), among the exponential number, that can be used to estimate the count for any pattern with minimal error. We envisage this information being made available as meta-data with each data set. In deference to the idea of a nutrition label, we call our stored information a *"label"*. An important feature of our model that is missing in previously proposed models for data labeling is the ability to generate the labels in a fully automated manner.

We define our notion of data labels with respect to a subset of attributes S, as the count information of all possible values combination of attributes in S appearing in the data. The size of the label is then determined by the space required for the count information. We present a model for pattern count estimation using a label. Given the estimation procedure, each label entails an error with respect to the real count of patterns in the data. The problem of finding an optimal label within a given bound on the label size is NP-hard (see [12] for details). A naive algorithm for the problem would traverse over all possible attribute subsets in increasing size order, compute the size of the corresponding label for each set, and choose the one that entails the minimal error within the space budget. We argue that in practice, the labels generated with a set of attributes S is preferable over labels generated using any subset of S, and build upon this property an optimized heuristic for the problem of finding an optimal label. Our experimental results demonstrate the high accuracy of the labels generated, even with a very limited space budget, and indicate the usefulness of our proposed optimized heuristic compared to the naive algorithm.

II. Labels and Pattern Count Estimation

In this section, we present a novel model of label construction, based on counts. We assume the data is represented using a single relational database, and that the relation's attributes values are categorical. Where attribute values are drawn from a continuous domain, we render them categorical by bucketizing them into ranges: very commonly done in practice to present aggregate results. In fact, we may even group categorical attributes into fewer buckets where the number of individual categories is very large. We first define the notion of pattern, which is the foundation for our label model.

Definition 2.1 (Patterns): Let D be a database with attributes $\mathcal{A} = \{A_1, \ldots, A_n\}$ and let $Dom(A_i)$ be the active domain of A_i for $i \in [1..n]$. A *pattern* p is a set $\{A_{i_1} = a_1, \ldots, A_{i_k} = a_k\}$ where $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \mathcal{A}$ and $a_j \in Dom(A_{i_j})$ for each A_{i_j} in p. We use $Attr(p)$ to denote the set of attributes in p.

Example 2.2: Consider the fragment of the simplified version of the COMPAS database [1] given in Figure 1. We use

	Gender	Age group	Race	Marital status
1	Female	under 20	African-American	single
2	Male	20-39	African-American	divorced
3	Male	under 20	Hispanic	single
4	Male	20-39	Caucasian	married
5	Female	20-39	African-American	divorced
6	Male	20-39	Caucasian	divorced
7	Female	20-39	African-American	married
8	Male	under 20	African-American	single
9	Female	20-39	Caucasian	divorced
10	Male	under 20	Caucasian	single
11	Male	20-39	Hispanic	divorced
12	Female	under 20	Hispanic	single
13	Female	20-39	Hispanic	married
14	Female	under 20	Caucasian	single
15	Female	20-39	Caucasian	married
16	Male	20-39	Hispanic	married
17	Male	20-39	African-American	married
18	Female	20-39	Hispanic	divorced

Fig. 1: Sample data from a simplified version of the COMPAS dataset

g, a, r and m as abbreviations for gender, age group, race and marital status. p ={g = female, a = 20-39, m = married} is a possible pattern and $Attr(p)$ ={g, a, m}.

Definition 2.3: We say that a tuple $t \in D$ *satisfies* a pattern p if $t.A_i = a_i$ for each $A_i \in Attr(p)$. The count $c_D(p)$ of a pattern p is the number of tuples in D that satisfy p.

Example 2.4: Consider again the database given in Figure 1 and the pattern p given in Example 2.2. The count of p is $c_D(p) = 3$ since the tuples 7, 13 and 15 satisfy it.

Information regarding the count of patterns appearing in the data can be useful to determine fitness for use. While the full count of each pattern provides a detailed and accurate description of the data, it can be extremely large. In fact, it can have the same size as the data. To this end, we propose an estimation function, which estimates a pattern count based on partial count information, we refer to as a label.

A label is defined with respect to a subset S of the database attributes, and it contains the pattern count (PC) for each possible pattern over S and value count (VC) of each value appearing in D. Given a subset of attributes $S \subseteq \mathcal{A}$ we use P_S to denote the set of all possible patterns over S (i.e., p with $Attr(p) = S$) such that $c_D(p) > 0$. The maximal number of patterns in P_S is $\prod_{A_i \in S} |Dom(A_i)|$.

Definition 2.5 (Label): Given a database D with attributes $\mathcal{A} = \{A_1, \ldots, A_n\}$, and a subset of attributes $S \subseteq \mathcal{A}$ a *label* $L_S(D)$ of D using S contains the set $PC = \{(p_i, c_D(p_i))\}$ for each $p_i \in P_S$ and the set $VC = \{(\{A_i = a_j\}, c_D(\{A_i = a_j\}))\}$ for each $A_i \in \mathcal{A}$ and $a_j \in Dom(A_i)$.

Example 2.6: Consider the database fragment given in Figure 1, the label resulting from use of the attributes set S = {a, m} consists of the following:

$PC = \{(\{$a = under 20, m = single$\}, 6),$
$(\{$a = 20-39, m = married$\}, 6),$
$(\{$a = 20-39, m = divorced$\}, 6)\}$
$VC = \{(\{$g = female$\}, 9), (\{$g = male$\}, 9), (\{$a = under 20$\}, 6),$
$(\{$a = 20-39$\}, 12), (\{$r = African-American$\}, 6),$
$(\{$r = Hispanic$\}, 6), (\{$r = Caucasian$\}, 6), (\{$m = single$\}, 6),$
$(\{$m = divorced$\}, 6), (\{$m = married$\}, 6)\}$

The label resulting from use of the attributes set S' = {g, a} consists of the same VC set and the following PC set:

$PC = \{(\{\text{g = female, a = under 20}\}, 3),$
$(\{\text{g = male, a = under 20}\}, 3), (\{\text{g = female, a = 20-39}\}, 6),$
$(\{\text{g = male, a = 20-39}\}, 6)\}$

Note that for a given database D, the VC set is determined and similar for every label of D.

Let D be a database with attributes $\mathcal{A}$, and S_1 and S_2 be two subsets of attributes such that $S_1 \subseteq S_2 \subseteq \mathcal{A}$. Given a pattern $p \in P_{S_2}$, we use $p|_{S_1}$ to denote the pattern that results when p is restricted to include only the attributes of S_1. Given $L_{S_1}(D)$, we may estimate the count of each pattern in P_{S_2}.

Definition 2.7 (Pattern Estimation): Let D be a database with attributes $\mathcal{A}$ and $S_1 \subseteq S_2 \subseteq \mathcal{A}$ be two subsets of attributes. Given a label $l = L_{S_1}(D)$ the count estimate for a pattern $p \in P_{S_2}$ is

$$Est(p, l) = c_D(p|_{S_1}) \cdot \prod_{A_i \in S_2 \setminus S_1} \frac{c_D(\{A_i = p.A_i\})}{\sum_{a_j \in Dom(A_i)} c_D(\{A_i = a_j\})}$$

Example 2.8: Consider again the database given in Figure 1, and the label $l = L_S(D)$ generated using S ={a, m} shown in Example 2.6. The estimate of the pattern p ={g = female, a = 20-39, m = married} using l is

$$Est(p, l) = c_D(\text{a = 20-39, m = married}) \cdot \frac{c_D(\{\text{g = female}\})}{\sum_{a_j \in Dom(\text{g})} c_D(\{\text{g} = a_j\})} = 6 \cdot \frac{9}{18} = 3$$

Using the label $l' = L_{S'}(D)$ generated from $S' = \{\text{g, a}\}$, with a similar computation we obtain

$$Est(p, l') = c_D(\text{g = female, a = 20-39}) \cdot \frac{c_D(\{\text{m = married}\})}{\sum_{a_j \in Dom(\text{m})} c_D(\{\text{m} = a_j\})} = 6 \cdot \frac{6}{18} = 2$$

We can then define the error of a label with respect to a pattern and a set of patterns.

Definition 2.9 (Estimation Error): The error of a label $l = L_S(D)$ with respect to a pattern p is

$$Err(l, p) = |c_D(p) - Est(p, l)|$$

Example 2.10: Reconsider the estimates $Est(p, l)$ and $Est(p, l')$ of the pattern p ={g = female, a = 20-39, m = married} shown in Example 2.8. The count of the pattern p in the database is 3, thus the error of l with respect to p is 0 and the error of l' is 1.

Abusing notation, we use $Err(l, \mathcal{P})$, for a set of patterns $\mathcal{P}$, to denote the maximum error in the estimate for any individual pattern in $\mathcal{P}$. We choose to focus on the maximum error (rather than mean for instance), as this definition of error is stiffer and gives us a sense of the error "bound" over a large number of patterns in the database. Our problem definition, its hardness and proposed solution holds also when using other error measures, such as $q\text{-}error$ [11], and we report the $q\text{-}error$ of our labels in the experiment in [12].

We are now ready to define the optimal label problem.

Definition 2.11 (Optimal Label Problem): Given a database D, with attributes $\mathcal{A}$, a bound B_s over the label size, and a set of patterns $\mathcal{P}$, the optimal label is

$$\arg\min_{S \subseteq \mathcal{A}} Err(L_S(D), \mathcal{P}) \text{ such that } |P_S| \leq B_s$$

Intuitively, the set of patterns $\mathcal{P}$ may be defined as $P_{\mathcal{A}}$ (i.e., the set of all possible patterns that include all the attributes and every value for each attribute that appears in the data). In this case $|\mathcal{P}| = |D|$ and an optimal label would be one that minimizes the error with respect to the count of tuples in the data. Our problem definition is more flexible, and allows the user to define a different pattern set, e.g., patterns that include only sensitive attributes.

To formally characterize the complexity of the optimization problem, we further need to define a corresponding decision problem. We define it as the problem of determining the existence of a label with size limited by the given bound and error which does not exceed a given error bound.

Definition 2.12 (Decision Problem): Given a database D, with attributes $\mathcal{A}$, a bound B_s over the label size, a set of patterns $\mathcal{P}$, and an error bound B_e, determine if there is a label $L_S(D)$ with $|P_S| \leq B_s$ and $Err(L_S(D), \mathcal{P}) \leq B_e$

We can show that (see proof in [12]).

Theorem 2.13: The decision problem is NP-hard.

III. Optimal Label Computation

Given a database D with attributes $\mathcal{A} = \{A_1, \ldots, A_n\}$ and a bound B_s, a naive algorithm for the optimal label computation would operate as follows: iterate over possible attributes sets, starting with sets of size 2. At each iteration, compute the set of all possible labels with a fixed size, namely, at the i'th iteration the algorithm generates the labels $\{L_{S_1}(D), \ldots, L_{S_k}(D)\}$, where each S_j for $j \in [1..k]$ is a subset of attributes of size $i + 1$. For each label generated, compute its size and error, and record the optimal label computed with size below the given bound. The algorithm terminates if the size of all the labels generated in the same iteration exceeds the bound (or when all possible subsets were generated). Intuitively, if every attribute subset of size i leads to a label with size greater than the given bound, then, every label generated using any attributes subset of size $> i$ would also exceed the bound. The naive algorithm is unacceptably expensive. Therefore we developed a much faster heuristic solution for the optimal label problem.

Label estimation characterization: We start by characterizing the count estimation for a given pattern using a given label. Let D be a database with attributes $\mathcal{A}$, $S \subseteq \mathcal{A}$ an attributes set and $l = L_S(D)$.

Definition 3.1: Given a pattern p, the estimate of p using l is an *exact* (*over* or *under*) *estimation* if $Est(l, p) = c_D(p)$ (resp., $Est(l, p) > c_D(p)$ or $Est(l, p) < c_D(p)$).

Clearly, for every pattern p if $Attr(p) \subseteq S$ then the estimate of p using l is an exact estimation. We can also show that:

Proposition 3.2: Given two attribute sets $S_1 \subseteq S_2 \subseteq \mathcal{A}$ and $l_i = L_{S_i}(D)$ the labels of D using S_i for $i = 1, 2$

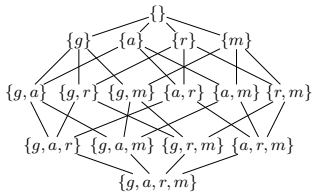

Fig. 2: A label lattice

respectively, for every pattern p such that $Attr(p) \not\subseteq S_2$ let $p' = p|_{Attr(p)\cap S_2}$ be the pattern resulting when restricting p to include only the attributes appearing in S_2. If the estimate of p' using l_1 is an over (under) estimation, and the estimate of p using l_2 is an over (resp., under) estimation then $Err(l_2, p) \leq Err(l_1, p)$.

Intuitively, for two attributes sets S_1 and S_2, if $S_1 \subseteq S_2$ the label generated using S_2 has more details than the one generated using S_1. In fact, based on Proposition 3.2, it is reasonable to assume that the pattern's count estimation using $L_{S_2}(D)$ is more precise than the one using $L_{S_1}(D)$. We show that this assumption indeed holds in practice in [12]

Our solution is based on the above observation, and is inspired by the Apriori algorithm [3] and the Set-Enumeration Tree for enumerating sets in a best-first fashion [13]. We start by defining a lattice over the possible labels, and then show how it can be used to compute the optimal label.

Definition 3.3 (Labels lattice): Given a database D with attributes $\mathcal{A}$, let $\mathcal{A}^*$ be the set of all possible subset of $\mathcal{A}$. The *label lattice* of D is a graph $G = (V, E)$, where $V = \mathcal{A}^*$ and $E = \{\{S_1, S_2\} \mid S_1 \subset S_2 \text{ and } \exists A_i \in \mathcal{A} \text{ s.t. } S_1 \cup \{A_i\} = S_2\}$.

S_1 is a parent (child) of S_2 if there is an edge $\{S_1, S_2\}$ and $S_1 \subset S_2$ ($S_2 \subset S_1$).

Intuitively, S_1 is a parent of S_2 if S_2 can be obtained from S_1 by adding a single attribute $A \in \mathcal{A} \setminus S_1$. Figure 2 depicts the label lattice of the database given in Figure 1.

We note that, due to the nature and purpose of the labels (i.e., conciseness that allow for user friendly visualization), the typical bound over the label size is small. Thus, a natural way to scan the lattice is from the top down. Traversing the lattice does not require explicit representation of the graph, as children nodes can be generated on demand from their respective parents. Moreover we can generate each node in the label lattice exactly once in a top down scan as we next show. To this end we define the operator $gen(S)$ for a subset of attributes S as follows.

Definition 3.4: Let D be a database with attributes $\mathcal{A} = \{A_1, \ldots, A_n\}$. We assume attributes are ordered, and for a given subset of attributes $S \subset \mathcal{A}$ we use $idx(S)$ to denote the index of the attribute with maximal attribute index in S, namely $idx(S) = \max_i(\{A_i \mid A_i \in S\})$, we define

$$gen(S) = \{S' \mid S' = S \cup \{A_j\} \; \forall j \text{ s.t. } idx(S) < j \leq n\}$$

The set $gen(S) \subseteq children(S)$ where $children(S)$ is the set of all children of S in the label lattice of D.

Example 3.5: For the attributes subset $S =$\{g, r\}, $gen(S)$ is \{g, r, m\}. Note that \{g, a, r\} is a child of S in the labels lattice, but is not included in $gen(S)$.

Algorithm 1: Top down search

input : A database D, a set of patterns $\mathcal{P}$ and a bound B_s.
output: Optimal label.

```
Q = [gen({})]
cands = ∅
while Q is not empty do
    curr ← Q.dequeue()
    for c ∈ gen(curr) do
        if labelSize(c, D) ≤ B_s then
            Q.enqueue(c)
            removeParents(cands, c)
            cands ← cands ∪ {c}
return L_S(D) for arg min_{S∈cands} Err(L_S(D), P)
```

Top down algorithm: Algorithm 1 finds the optimal label using a top down traversal of the label lattice. The algorithm gets as input a database D, a set of patterns, and a bound B_s. It uses a queue Q to generate a candidate list of attributes subset, $cands$, such that the size of the label generated using each candidate in the list does not exceed the bound B_s. By traversing the lattice in a top down fashion using the gen operator the algorithm generates each node in the lattice at most once. Furthermore, the nodes generated are only attribute sets that lead to labels with size below the given bound, and (in the worst case) their children.

Proposition 3.6: Given a database D, a set of patterns $\mathcal{P}$ and a bound B_s, Algorithm 1 generates each node in the label lattice at most once.

Algorithm 1 avoids generating and exploring a large portion of the labels lattice, and in particular most of the labels that exceed the bound limit (which are the majority in practice, see Section IV-B).

IV. Experimental Evaluation

We conducted experiments on real data to assess the quality of our proposed labels in estimating the data pattern's count. The key concerns are the label's size and the error in estimation. We evaluated this trade-off and considered the impact of data set parameters. We compared our label's accuracy to the performance of a real DBMS estimator, and the conventional approach of sample-based estimation using different error measures. A second issue we studied is the performance of the label generation algorithm. We examined scalability in terms of label generation time as a function of (i) label's size bound, (ii) data size, and (iii) number of data attributes. We also quantified the usefulness of the heuristic approach compared to the naive algorithm. In this section, we report part of these experimental results. The full detailed evaluation is given in [12]. We begin with the set up we used.

A. Experimental setup

We used three real datasets with different numbers of tuples and attributes as follows.

Blue Nile: An online jewelry retailer. We used the dataset collected and used in [5] of diamonds catalog, containing 116,300 diamonds. The dataset has 7 categorical attributes.

The COMPAS dataset: Collected and published by ProPublica [1]. It contains 60,843 records that includes demographics, recidivism scores, and criminal offense information. The total number of attributes in the original database was 29. We removed id attributes (person id, assessment id, case id), names (first, last and middle), dates and attributes with less than 2 values or over 100 values. We added the attribute age, with four age ranges, based on the date of birth attribute. The resulting dataset contains 17 attributes.

Default of Credit Card Clients Dataset [2]: This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. It has 24 attributes and 30,000 tuples. We bucketize each numerical attribute into 5 bins.

In all the experiments we set $\mathcal{P}$, the patterns set, to be $P_{\mathcal{A}}$ where $\mathcal{A}$ in the set of all attributes in the dataset; namely, the set of possible patterns that include all the attributes and every value for each attribute that appears in the data. The experiments were executed on macOS Catalina, 64-bit, with 16GB of RAM and Intel Quad-Core i7 3.1 GHz processor. All algorithms were implemented in Python 3.

B. Experimental results

Label accuracy: We assessed the quality of the generated labels in estimating the data pattern's count by examining the error induced by the labels of varying size with respect to the set of patterns appearing in the database. We varied the label's size bound from 10 to 100 to generate labels with different size. Error was measured as the absolute value of difference in count between the actual and estimated count for each pattern. In [12] we report the q-error of the estimations.

We have compared the accuracy of our proposed pattern count based label (PCBL, blue line in the graphs) to two baseline approaches: (1) the PostgreSQL row estimation that relies on 1D histograms. It stores statistical data about the database in `pg_statistic` and random sampling while producing statistics. (2) Uniform random sample with growing size. The size of a sample that corresponds to the bound x is $x + |VC|$. Given a sample S of size $|S|$ for a dataset D, and a pattern p, we use $c_S(p) \cdot \frac{|D|}{|S|}$ to estimate the count of p in D, where $c_S(p)$ is the count of p in S.

For all three datasets, we observed similar errors for the label generated by the optimal heuristic and the one generated by the naive algorithm (blue line in the graphs). In all cases `pg_statistic` contained over 400 rows (429 in the BlueNile dataset, 439 in the COMPAS dataset, and 446 in the Credit Card). The accuracy is independent of the label size, and is marked with a gray line in the graphs. For the sample based estimation we report the average over 5 executions and the results are marked in yellow.

Figure 3 shows the absolute max error (mean error values are shown in parenthesis) as a function of the label size. The maximal error is presented as a fraction of the data size. For the BlueNile dataset, in the sample based estimation we observed a small increase in the maximal error for a sample of 75 (corresponds to label with $|PC| = 28$, bound of 30). This is because the sample size is significantly smaller that the database size, thus $\frac{|D|}{S}$ is larger that the count of all tuples in the data, which results in over estimation for all tuples in the sample, and estimation of 0 for the rest. In particular, if the count of a pattern is greater than 2 (as in one of the executions in this experiment) the overestimation is even higher. The mean error of the sample based method decreased from 18.44 for the smallest sample size ($\times 3$ of the PCBL) to 17.04 in the largest sample (over $\times 4$ of the PCBL).

In the Credit Card dataset, for a label with 92 pattern-count pairs (generated with the bound set to 100), we obtain maximum error of 607 (2.0%). The maximum observed error remains 607 when we increased the label size bound from 70 to 100 (generating labels of size 70 and 92 respectively). We note that the mean error decreased 2.2978 to 2.2974. To further demonstrate the trend, we examine the error of labels generated with bound set to 125 and 150, which generated labels of size 121 and 139 respectively. The maximal error of the average sample based estimation decreased from 789 to 453, which is slightly better than the results of the PCBL, however the mean error was higher.

Label generation time: The next set of experiments aims at studying the scalability of the algorithms for label generation. We compared the performance of our proposed optimized heuristic algorithm (dark blue) to a baseline naive algorithm described in Section III (light blue). Figure 4 depicts the running time as a function of the label's size bound from bound 10 and up to 100. As the bound grows, the number of possible attributes subsets that may be used to generate an optimal label increases, which affect the generation time for both algorithms. The optimized heuristic outperform the naive algorithm since the number of subsets it consider is smaller. In the Credit Card dataset, the naive algorithm did not terminate within 30 minutes beyond bound of 50. For bound of 50 the naive algorithm running time was over 18 minutes. The optimal heuristics was able to compute the label for bound of 50 with about 3.5 minutes, and the label for the largest bound of 100 within 18 minutes.

Effect of optimization: Recall that our heuristic optimizes the number of attribute sets examined during the search of the optimal label. To quantify the usefulness of our heuristic, we compared the number of attributes sets examined during the label generation by the optimized heuristic and the naive algorithm. We observed a gain of up to 99% in the number of subsets examined as shown in Figure 5.

V. Related Work

With increasing interest in data equity in recent years, multiple lines of work have focused on labeling data and models in order to improve transparency, accountability and fairness in data science [7], [9], [10], [14]–[16].

Different data labeling models were studied in [7], [9], [15]. Data nutrition labels [9] are composed of modules, called widgets. Modules are stand-alone, and each provides

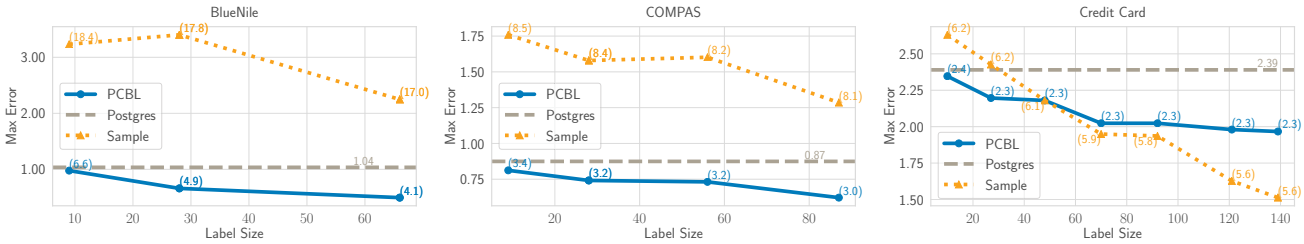

Fig. 3: Absolute max error as a function of label size (mean values are shown in parenthesis)

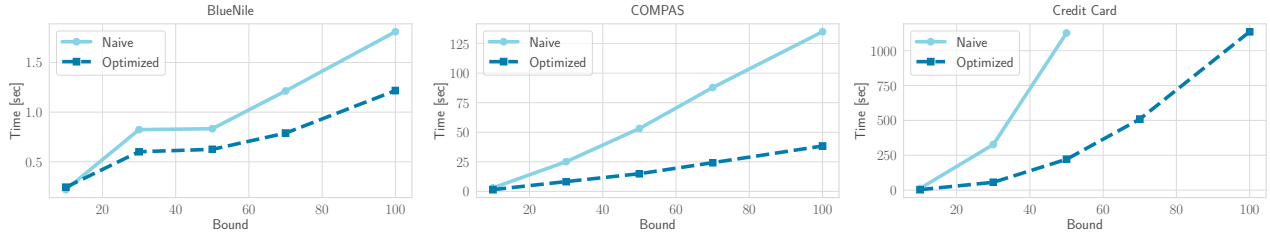

Fig. 4: Label generation runtime as a function of label size bound

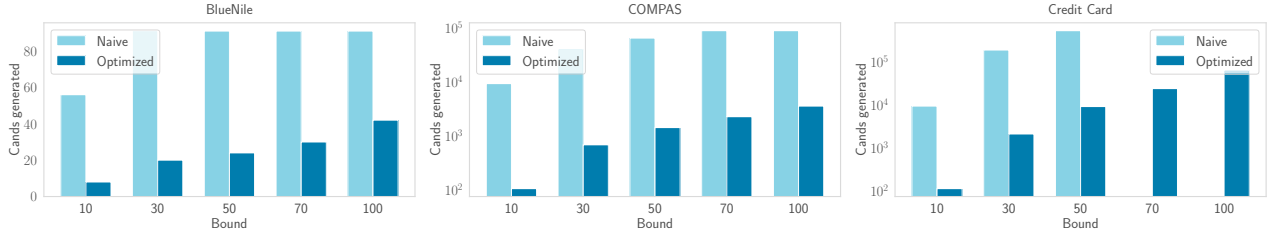

Fig. 5: Number of labels candidates examined as a function of label size bound

a different flavor of information: metadata, provenance, variables, statistics pair, probabilistic model and ground truth correlations. The models vary in the manual effort required for their generation and their technical sophistication. Overall, the labels allow users to interrogate various aspects of the dataset. Our proposed label model may be assimilated as a widget or a module in the above models. Other works [10], [16] have focused on model labeling.

While the idea of a nutritional label has been very nicely argued for in work such as that cited above, the actual content of the label is either manually generated, or at most has an aspiration towards automated generation beyond the simplest properties. Our work establishes the first critical widget that provides substantive information about a data set and is constructed in a completely automated manner.

VI. Conclusion

We have developed a "label" for a dataset that can be used to determine the count for every pattern in the dataset. These counts are typically central to determining fitness for use, and thus avoid generating biased models and data-driven algorithms. Our work is in line with the many recent proposals for a data set label that allows users to determine fitness for use and build trust.

References

[1] Compas recidivism risk score data and analysis. https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis.

[2] Default of credit card clients data set. https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*. Morgan Kaufmann, 1994.

[4] Julia Angwin, Jeff Larson, Lauren Kirchner, and Surya Mattu. Machine bias, May 2016.

[5] Abolfazl Asudeh, Zhongjun Jin, and H. V. Jagadish. Assessing and remedying coverage for a given dataset. In *ICDE*, 2019.

[6] Irene Y. Chen, Fredrik D. Johansson, and David A. Sontag. Why is my classifier discriminatory? In *NeurIPS*, 2018.

[7] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *CoRR*, abs/1803.09010, 2018.

[8] Jindong Gu and Daniela Oelke. Understanding bias in machine learning. *CoRR*, abs/1909.01866, 2019.

[9] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards. *CoRR*, abs/1805.03677, 2018.

[10] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *FAT**, pages 220–229. ACM, 2019.

[11] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proc. VLDB Endow.*, 2(1), 2009.

[12] Yuval Moskovitch and H. V. Jagadish. Patterns count-based labels for datasets [technical report]. *CoRR*, abs/2010.16340, 2020.

[13] Ron Rymon. Search through systematic set enumeration. In *KR*. Morgan Kaufmann, 1992.

[14] Julia Stoyanovich and Bill Howe. Nutritional labels for data and models. *IEEE Data Eng. Bull.*, 42(3), 2019.

[15] Chenkai Sun, Abolfazl Asudeh, H. V. Jagadish, Bill Howe, and Julia Stoyanovich. Mithralabel: Flexible dataset nutritional labels for responsible data science. In *CIKM*. ACM.

[16] Ke Yang, Julia Stoyanovich, Abolfazl Asudeh, Bill Howe, H. V. Jagadish, and Gerome Miklau. A nutritional label for rankings. In *SIGMOD*. ACM, 2018.