DOI:10.1145/3500923

Data-centric methods designed to increase end-to-end reliability of data-driven decision systems.

BY YUVAL MOSKOVITCH AND H.V. JAGADISH

Reliability at Multiple Stages in a Data Analysis Pipeline

THE UBIQUITY OF data in recent years has led to wide use of automated, data-driven decision-making tools. These tools are gradually supplanting humans in a vast range of application domains, including decisions about who should get a loan,[a] hiring new employees,[b] student grading,[7] and even assessing the risk of paroling convicted criminals.[4] Our growing dependence on these tools, in particular in domains where data-driven algorithmic decision making may affect human life, raises concerns regarding their reliability. Indeed, with the increasing use of data-driven tools, we also witness a large number of cases where these tools are biased. For instance, COMPAS, a risk assessment tool that predicts the likelihood of a defendant re-offending, was widely used in courtrooms across the U.S. *ProPublica*, an independent, nonprofit newsroom that produces investigative journalism in the public interest, conducted a study on COMPAS, which showed the software discriminated based on race. Black people were scored as a greater risk to re-offend than the actual, while White people were scored as a lower risk than the actual.[4] Further analysis[5] revealed issues with other groups as well. For example, the error rate for Hispanic women is very high because there are not many Hispanic women in the dataset. It is not only that there are fewer Hispanics than Black and White people, and fewer women than men, but also fewer Hispanic women than one would expect if these attribute values were independently distributed.

Another example, recently published in *The New York Times*,[7] showcases a different bias scenario. The International Baccalaureate (IB) is a global standard of educational testing that allows U.S. high-school students to gain college credit. The final exams, a major factor in student scores, were canceled due to the COVID-19 pandemic. Instead, students were assigned grades based on

» key insights

- **Data-driven methods are increasingly being used in a wide range of application domains, such as hiring, grading, and risk assessment, where data-driven algorithmic decision making may affect human life. With the increasing use of such tools, we also witness many cases where they are biased and cause harm.**
- **The development of data-driven decision systems is typically complex and consists of numerous phases. Bias may be introduced at different points in the development pipeline.**
- **The growing impact of data and data-driven systems on society brings out the need for reliable data management and analysis methods. With the complexity of those systems, multiple tools must be used together to construct data-based decision systems with end-to-end reliability.**

IMAGE BY OMELCHENKO

a https://www.forbes.com/sites/parmyolson/2011/03/15/the-algorithm-that-beatsyour-bank-manager/?sh=4fbafadc1ae9

b https://www.nytimes.com/2015/06/26/upshot/can-an-algorithm-hire-better-thana-human.html

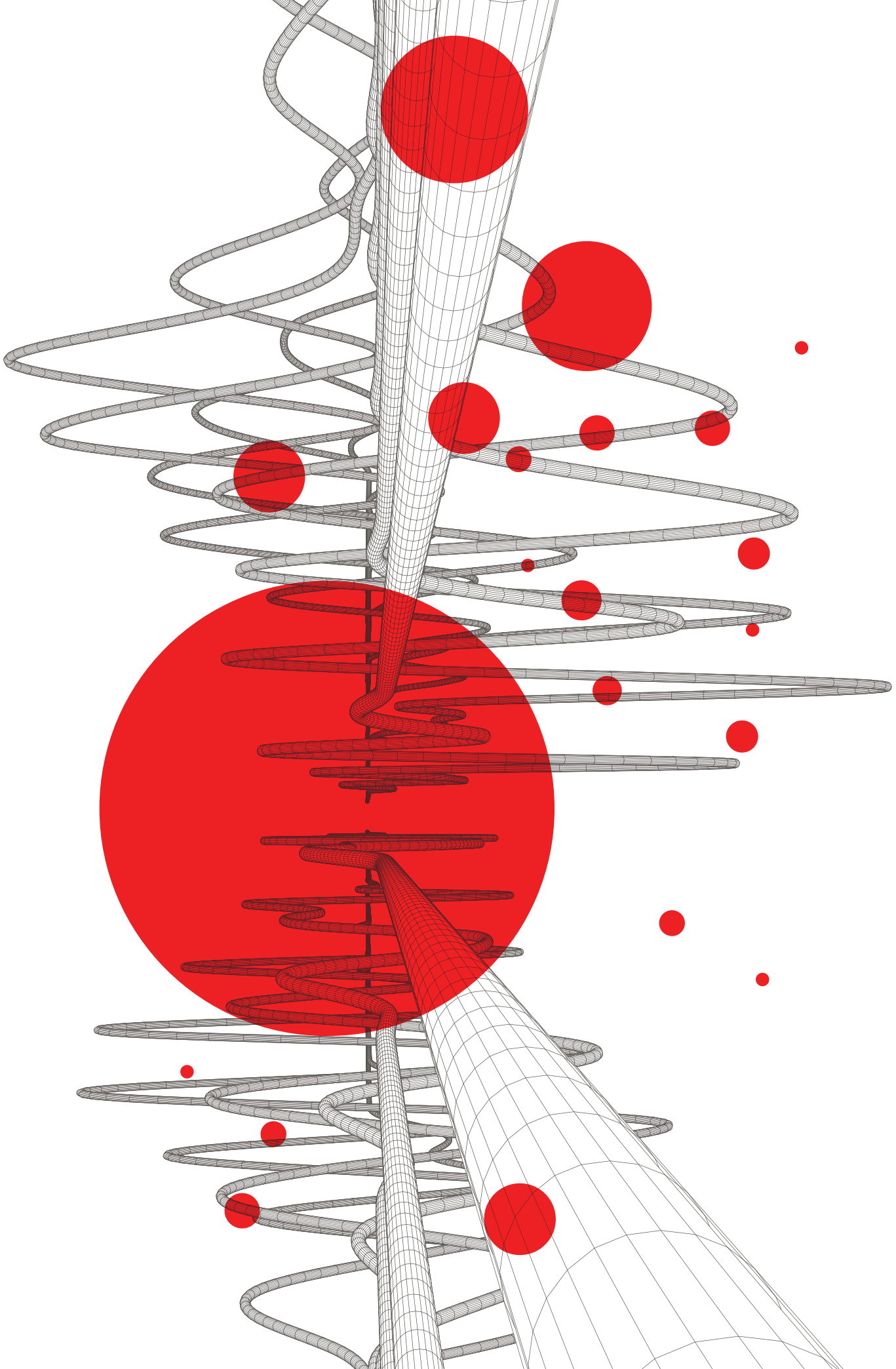
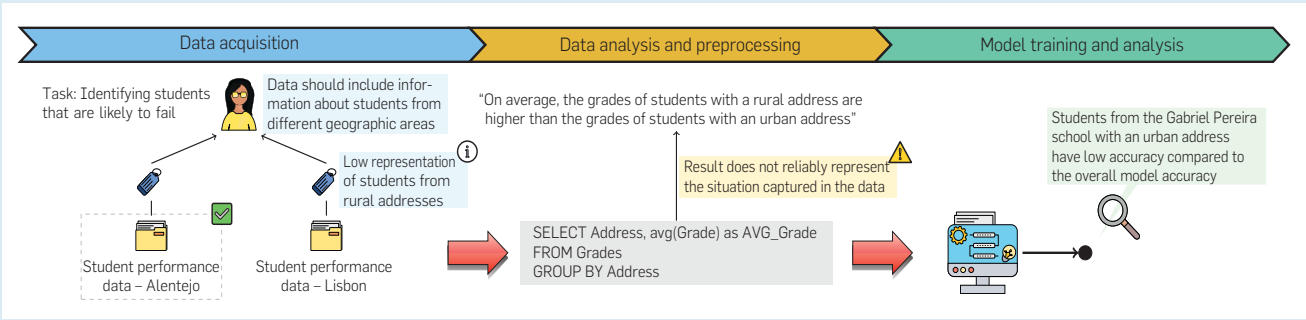


Figure 1. Data-driven decision systems' development pipeline: generating decision-making tools from raw data. The pipeline begins with data acquisition and consists of data preprocessing and ML model training. Our proposed methods for increasing end-to-end reliability include data labeling, which allows users to determine the fitness of data representation for the application; aggregation results quality assessment; and detection of unfairly treated groups.



a predictive model. As a result, high-achieving students from poor school districts were severely hurt, because the model placed great weight on school quality. For example, students from low-income families were predicted to fail the Spanish exam, even when they were native Spanish speakers. Many of them had studied for the IB hoping to save thousands of dollars on tuition by earning college credit with their scores.

Erroneous decisions may also be caused by generalizing from detailed data to statements in a broader context. Aggregation queries (that is, queries that group multiple data items and form a single summary value, such as average) can be used to generate such statements. Statements based on aggregation query results over datasets are commonly used by data scientists when analyzing large datasets. These statements, which we refer to as *generalizations*, allow analysts to achieve and to convey a high-level understanding of the data. Poorly constructed generalizations might convey misleading information even if the statements are technically supported by the data. Misleading statements could be made intentionally—for example, by using some highly influential data points, namely data points with high variance values (such as outliers). This affects an entire aggregated group result when performing aggregation. Examples can be found in many statements made by politicians, but they can even occur in "objective" arenas, such as science and medicine. For instance, doctors have over-diagnosed ADHD for years after making generalizations to age, sex, and the maturity of the children.[19]

The need to address such concerns has been felt by many. There is a significant body of literature on fairness in AI,[21] on estimating the reliability of statements representing a document or dataset,[11] and so on. However, out of necessity, most such works have addressed a particular narrow concern. The challenge for a data-science pipeline is that it has to establish end-to-end reliability. Reliability is a wide concept that encompasses numerous aspects. In particular, there are many facets in which the reliability of data-driven decision systems can be improved along its development process. The following example demonstrates potential challenges and pitfalls in the development process.

Example 1. Consider an initiative to improve the educational level in Portugal by identifying students likely to fail core classes. Providing these students with academic assistance at an early phase could potentially increase their success rate. Figure 1 depicts the development process of a data-driven decision system for identifying students who need support.

The process starts with data collection. Multiple datasets with information on student performance are available. For instance, "The Student Performance Data Set"[8] features data collected from two schools in Portugal's Alentejo region: Gabriel Pereira (GP) and Mousinho da Silveira (MS). The data was collected during the 2005–2006 school year and contains the performance of 1,044 students in the Math and the Portuguese language exams, along with demographic, social/emotional, and school-related information. A similar dataset is also available from the capital Lisbon.

The data scientist then needs to decide which data should be used for the development of the system. Since the system is developed as part of a national effort, the data should represent students from different geographic areas. Exhausting and time-consuming profiling of the two datasets reveals insufficient representation of students from rural addresses in the data collected from schools in Lisbon. Keeping in mind that low representation of students from rural addresses may lead to poor performance of the resulting system with respect to this group, the data scientist selects the Alentejo region data.

In analyzing the data, the scientist executes queries over the dataset to gain a better understanding of the data. One of the queries computes the average grades of students based on their address. According to the query result, the data scientist concludes that the grades of students with a rural address are higher than the grades of students with an urban address. While this statement is true, it is in fact misleading. Closer examination of the data reveals a big disparity in trends based on sex: The grades of female students from urban addresses are higher, but the grades of male students from rural addresses are higher. Pooling these two together gives rural students a small lead in the aggregate, but that misses the key trends.

Finally, the data scientist uses the data to train a machine-learning (ML) model. In the analysis of the resulting

model, the user observes a satisfactory overall accuracy. To further verify the model's fairness, the scientist computes the accuracy of different groups in the data: male and female, students from urban and rural addresses, and any combination thereof. As the accuracy of all groups is comparable, the model is deployed in the system. When using the system, users in some of the schools encountered high error rates among students with an urban address. As a result, students who needed assistance were not identified by the system, while others, who were able to succeed without help, were given extra support. A reexamination of the ML model reveals that, indeed, due to the use of historical results of the schools, students from the Gabriel Pereira school who live in an urban address suffer from a significantly low accuracy compared with the model's overall accuracy.

We next outline our proposed solutions to assist users in the development process and eliminate potential bias.

Fitness of data representation to the application. When a decision is made by a machine-learned model, the quality of the decision depends centrally on the data used in the model training phase. For instance, as shown in Asudeh et al.,[5] insufficient group representation may lead to poor performance with respect to the group. Thus, our first approach focuses on the selection of training data and, particularly, on the representation of different groups in the data to help users determine the fitness of the data for the intended application. For instance, in Example 1, the representation of students from different geographic areas is crucial to the task. Different tasks may imply other representation requirements. For example, the Lisbon dataset may be a better fit for a program by the Lisbon Municipality aimed at advancing women in STEM. In this case, the number of females is also an important feature in selecting the dataset.

Detection of unfairness. The second method considers the fairness measures of the model. Data-driven tools that seemingly work well in general may treat minority groups in an unfair manner. The common approach in fairness analysis assumes a given set of sensitive attributes, which defines groups that may be treated unfairly. Following this assumption, an analyst may miss out on groups that are defined using attributes that are not naturally considered as "sensitive," as demonstrated in the IB example, where the school ID is used to define the unfairly treated group. Without a deep understanding of the domain, we would not know to choose a combination of school ID and home address as a sensitive attribute. To this end, we present a technique for detecting groups that are treated unfairly by a given ML model. In other words, we want to let the data speak to (potential) unfairness without requiring a human modeler to identify protected attributes ahead of time.

Aggregation-result quality assessment. Aggregation queries are commonly used by analysts throughout the analysis of large datasets, the development of data-driven decision tools, and in the decision-making pipeline. Analysts use aggregation queries to gain an understanding and a high-level description of the data. Evaluating the "representability" quality of an aggregate query result is crucial to detecting misleading conclusions. We thus propose a model to quantify the quality of generalizations derived by aggregate queries.

We will demonstrate the ideas presented in the paper using the "The Student Performance Data Set."[8] Figure 2 depicts a sample from the data with the attributes: gender, school, address (urban or rural), and failures (the number of past class failures). The grade attribute depicts the student grades on a scale from 0–20, and the prediction attribute describes the prediction, "pass" (grade $>$ 10) or "fail" (grade $\leq$ 10) of a predictive model.

Reliable Datasets

The data on which data-driven decision systems depend, as is often the case in data science, is typically "found data." That is, the data was not collected as part of the development of the analytics pipeline but was acquired independently, possibly assembled by others for different purposes. When using "found data," analysts typically perform data profiling, a process of extracting metadata or other informative summaries of the data.[2] While informative and useful, data profiling is hard to do well, is usually not automated, and requires significant effort. Instead, inspired by the notion of a "nutrition label,"[20] where the basic idea is to capture dataset properties of interest in a succinct label, we propose[14,15] labeling datasets with information regarding the count of different groups (defined using value combinations) in the data. Needless to say, there are a number of such combinations possible. So, storing individual counts for each is likely to be impossible. To this end, we focus on techniques to estimate these counts based on storing only a limited amount of information.

Given a dataset, if we do not know anything about its value distributions, a common assumption to make is

Figure 2. Student grades and ML prediction model for sample data from the "Student Performance Data Set." [8] The ML prediction is wrong for the highlighted rows (false positive in green, and false negative in red).

#	Gender	School	Address	Failures	Grade	Prediction
1	F	MS	R	1	11	Pass
2	M	MS	R	1	14	Pass
3	M	GP	U	1	9	Pass
4	M	GP	U	2	7	Fail
5	M	MS	R	0	19	Pass
6	F	MS	U	1	7	Fail
7	F	GP	R	1	9	Fail
8	M	GP	R	1	8	Fail
9	F	MS	R	0	12	Pass
10	F	MS	R	2	9	Fail
11	M	MS	R	2	12	Pass
12	F	GP	U	0	19	Pass
13	F	GP	U	2	12	Fail
14	M	MS	U	1	12	Pass
15	F	GP	U	1	8	Fail
16	M	GP	U	0	9	Fail

that of independence between attributes. Then, we can keep counts for only individual attribute values, and estimate counts for attribute value combinations, assuming independence. 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 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 impacts of race and gender alone. In other words, we have to ensure that our estimates for the count of any group in the database are at least approximately correct.

Histograms have long been used for similar purposes in relational databases;[9] however, they do not do very well in high dimensions. Other prevalent techniques for selectivity estimation include sampling[16] and machine learning-based methods.[22] The former suffers from insufficient performance in the presence of skews and high-selectivity queries, and the latter requires training and results in very complex models. Following the concept of nutrition labels for datasets, a key requirement in our problem context is that the metadata annotation can be immediately comprehensible to a potential user of the dataset.

We enhance user understanding of the appropriateness of datasets, and hence their willingness to trust them, through the effective use of dataset labels. We start by presenting our model of label construction, based on counts. We assume the data is represented using a single relational database, and that the relation's attribute values are categorical. Where attribute values are drawn from a continuous domain, we render them categorical by bucketing 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.

Automated, data-driven decision-making tools are gradually supplanting humans in a vast range of application domains.

Patterns count. Given a database D with attributes $\mathcal{A} = \{A_1, \ldots, A_n\}$, A we use $Dom(A_i)$ to denote 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. We say that a tuple $t \in D$ *satisfies* the 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 then the number of tuples in D that satisfy p.

Example 2. Consider the dataset in Figure 2. p = {Gender = M, School = MS, Address = R} is an example of a pattern. The tuples 2, 5, and 11 satisfy it and thus $c_D(p) = 3$.

While useful, storing count information of the patterns appearing in the data is likely to be impossible as their number is exponential in the number of attributes. We next present our method for estimating these counts using only a limited amount of information.

Pattern count-based labels. 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 metadata with each dataset. 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 labels in a fully automated way. We define our notion of data labels with respect to a subset of attributes S, as the count information of all possible value combinations of attributes in S in the data. The size of the label is then determined by the space required for the count information. More formally, a label $L_S(D)$ of D defined with respect to a subset S of the database attributes contains the following: (1) the pattern count (*PC*) for each possible pattern over S (that is, p with $Attr(p) = S$), and (2) value count (*VC*) of each value appearing in D.

Example 3. Consider the dataset in Figure 2. The label resulting from use of the attributes set S = {School, Address}

consists of the following:[c]

$$PC = \{(\{\text{School} = \text{GP}, \text{Address} = \text{U}\}, 6),$$
$$(\{\text{School} = \text{GP}, \text{Address} = \text{R}\}, 2),$$
$$(\{\text{School} = \text{MS}, \text{Address} = \text{U}\}, 2),$$
$$(\{\text{School} = \text{MS}, \text{Address} = \text{R}\}, 6),$$
$$VC = \{(\{\text{Gender} = \text{M}\}, 8), (\{\text{Gender} = \text{F}\}, 8),$$
$$(\{\text{School} = \text{GP}\}, 8), (\{\text{School} = \text{MS}\}, 8),$$
$$(\{\text{Address} = \text{U}\}, 8), (\{\text{Address} = \text{R}\}, 8),$$
$$(\{\text{Failures} = 0\}, 4), (\{\text{Failures} = 1\}, 8),$$
$$(\{\text{Failures} = 2\}, 4), (\{\text{Grade} = \text{Pass}\}, 8),$$
$$(\{\text{Grade} = \text{Fail}\}, 8)\}$$

The label resulting from use of the attributes set $S' = \{\text{Gender, School}\}$ consists of the same *VC* set and the following *PC* set:

$$PC = \{(\{\text{Gender} = \text{M}, \text{School} = \text{GP}\}, 4),$$
$$(\{\text{Gender} = \text{M}, \text{School} = \text{MS}\}, 4),$$
$$(\{\text{Gender} = \text{F}, \text{School} = \text{GP}\}, 4),$$
$$(\{\text{Gender} = \text{F}, \text{School} = \text{MS}\}, 4)\}$$

Count estimation. Given a database D with attributes $\mathcal{A}$ and a subset of attributes $S \subseteq \mathcal{A}$, we use P_S to denote the set of all possible patterns over S such that $c_D(p) > 0$. Let 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 in S_1. Given a label $l = L_{S_1}(D)$ of D using S_1, we may estimate the count of each pattern in P_{S_2} as follows.

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

Example 4. Consider again the dataset Figure 2, and the label $l = L_S(D)$ generated using $S = \{\text{School, Address}\}$ shown in Example 3. The estimate of the pattern $p = \{\text{Gender} = \text{M}, \text{School} = \text{MS}, \text{Address} = \text{R}\}$ using l is

$$Est(p, l) = c_D(\text{School} = \text{MS}, \text{Address} = \text{R}) \cdot \frac{c_D(\{\text{Gender} = \text{M}\})}{\sum_{a_j \in Dom(\text{Gender})} c_D(\{\text{Gender} = a_j\})} = 6 \cdot \frac{8}{16} = 3$$

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

$$Est(p, l') = c_D(\text{Gender} = \text{M}, \text{School} = \text{MS}) \cdot \frac{c_D(\{\text{Address} = \text{R}\})}{\sum_{a_j \in Dom(\text{Address})} c_D(\{\text{Address} = a_j\})} = 4 \cdot \frac{8}{16} = 2$$

Given the estimation procedure, each label entails an error with respect to the real count of patterns in the data. We define the error of a label $l = L_S(D)$ with respect to a pattern p as

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

Example 5. Reconsider the estimates $Est(p, l)$ and $Est(p, l')$ of the pattern $p = \{\text{Gender} = \text{M}, \text{School} = \text{MS}, \text{Address} = \text{R}\}$ shown in Example 4. 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 $\mathcal{P}$. The problem of finding an optimal label within a given bound on the label size is NP-hard (see our previous work[15] for details). This problem resembles the problem of query optimization with materialized views,[6] where the goal is to find a set of views that could be used to optimize query evaluation time, when the number of the views is limited and their maintenance under data updates is taken into account. In contrast, in our problem, we search for a label that would result in the most accurate estimations.

Optimal label computation. We next present our solution for optimal label computation. 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 minimal error within the budgeted space.
The naive algorithm is unacceptably expensive; therefore, we developed a much faster heuristic solution[15] for the optimal label problem. Our algorithm builds upon two observations. First, due to the nature and purpose of the labels (that is, conciseness that allows for user-friendly visualization), the typical bound over the label size is small. Second, we argue that in practice, labels generated with a set of attributes S are preferable over labels generated using any subset of S. Intuitively, for two attribute 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.

Our solution is inspired by the Apriori algorithm[3] and the Set-Enumeration Tree for enumerating sets in a best-first fashion.[17] We use a lattice over the set of all possible subsets of attributes, each corresponding to a possible label, such that labels located higher in the lattice are smaller. Following the first observation (the typical bound is small), we traverse the lattice in a top-down fashion. Traversing the lattice does not require explicit representation of it, as children nodes can be generated on demand from their respective parents. Moreover, each node is generated at most once in the scan. For each node generated by the algorithm, we compute the corresponding label size and prune the search based on the given label bound. Finally, the algorithm computes the error of potential labels observed during the search. The error computation of a label is time-consuming; thus, to optimize the performance of the algorithm, based on the second observation, the algorithm computes the error only for maximal sets. The label with minimal error is then returned. Our experiments 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 with the naive algorithm (see our previous work[15] for details).

Example 6. Figure 3 shows a label of "The Student Performance Data Set"[8] (including only data in Math) with the label size (the PC set) limited to 4. The label was generated using the "School" and "Address" attributes and includes information on the estimations error with respect to the patterns in the data (average error, maximal error, and standard deviation). Some immediate observations that can be made based on this information are that there is a high representation of students with urban addresses compared with students

c The grade attribute is bucketized into two bins: Pass (>10) and Fail (≤10).

from rural areas. Additionally, due to the low number of students who get extra paid classes, their number is likely to be insufficient for the development of a non-biased algorithm using this data. Note that even with a size limit as low as 4 the label significantly improves count estimations. For instance, assume that we are interested in the number of students from the GP school with a rural address and who do not pay for extra classes. Their true number in the dataset is 72. Using the attribute independence assumption, the count estimation is $649 \cdot \frac{423}{649} \cdot \frac{197}{649} \cdot \frac{610}{649} = 120$, whereas using the label we get a much more accurate estimation of $78 \cdot \frac{610}{649} = 73$.

Reliable Fairness Measures

Fairness is a crucial aspect of decision-making tool reliability. Once the ML model is trained, data analysts may wish to validate its fairness toward different groups in the data. Various fairness measures have been proposed, where different measures are typically designed to capture case-appropriate properties: Different definitions may be used in different use cases. For instance, one natural fairness definition considers the accuracy among different groups. According to this definition, a classifier is fair if different groups in the data (that is, Hispanic students from low-income families) have the same overall accuracy in prediction as other groups. Another plausible definition takes into account the false-positive error rates. This definition is appropriate when we wish to minimize the error of falsely classifying subjects in the negative class as positive—for example, granting loans to people who would not be able to pay back. Whether a classifier is fair depends on the notion of fairness. Different definitions can lead to different outcomes as we next show.

Example 7. Consider the dataset in Figure 2 and the classifier whose results are depicted in the "Prediction" column. The classifier's overall accuracy (based on the given data) is 0.875; however, for students from the GP school with an urban address, the accuracy is only 0.67, significantly lower than any other group in the data (that may be defined by values of the different attributes). Another fairness measure, known as equal opportunity, focuses on the false-positive error rate ($FPR = \frac{FP}{FP+TN}$). Intuitively, by this measure the classifier should produce similar results for all students who fail the exam. In this case, a high *FPR* indicates fairness issues. In our example, male students with a single previous failure have a high *FPR* (0.5) compared with the overall *FPR* (0.11).

There is extensive work on fairness, but the typical formulation works with explicitly pre-identified "protected" groups identified by particular values in specified attributes. For example, women can be a protected group based on a gender attribute. While it is important to work with protected groups due to historical discrimination based on race, gender, and ethnicity, these are not the only groups that can suffer from unfairness. Furthermore, as individuals, each of us is identified with many different groups.

Several recently developed tools (see, for example, the AI Fairness 360 project[1]) allow users to assess fairness of ML systems and even assist in mitigating the bias and provide explanations. However, they focus on investigating only user-specific protected groups. To build a reliable system, we must detect unfairness for all groups, including intersectional groups. Once such groups are detected, the user can remedy the problem, either by adjusting the ML model or altering the training data. We start by presenting our method[12] to do this for the problem of detecting groups with low accuracy and then present a generalized solution for other fairness definitions.

Figure 3. A label computed for The Student Performance Data Set[8] (using only data in Math) with the size limited to 4. Note: the PC set is only the last four rows of the table; the top 20 rows are the marginal distributions of individual attributes, which are basic statistics about this dataset.

Total size: 649

Attribute	Value	Count	
School	Gabriel Pereira	423	**65%**
	Mousinho da Silveira	226	**35%**
Gender	Male	266	**41%**
	Female	383	**59%**
Address	Urban	452	**70%**
	Rural	197	**30%**
Failures	0	549	**85%**
	1	70	**11%**
	2	16	**2%**
	3 or more	14	**2%**
School support	Yes	68	**10%**
	No	581	**90%**
Family support	Yes	398	**61%**
	No	251	**39%**
Extra paid classes	Yes	39	**6%**
	No	610	**94%**
Absences	0	244	**37%**
	1	12	**2%**
	2	110	**17%**
	3 or more	283	**44%**

School	Address	Count	
Gabriel Pereira	Urban	345	**53%**
	Rural	78	**12%**
Mousinho da Silveira	Urban	107	**17%**
	Rural	119	**18%**
Average Error	1.08	**0.17%**	
Maximal Error	9	**1.39%**	
Standard deviation	1.8		

Detecting groups with low accuracy. Given a classifier M and a labeled dataset D, the *accuracy* of a pattern p, denoted as $acc_M(p)$, is the ratio of the number of tuples in D satisfying p that are correctly classified by M to $c_D(p)$.

Example 8. Consider again the dataset in Figure 2. The accuracy of the pattern p = {School=GP, Address=U} is $acc_M(p) = \frac{4}{6} = 0.67$, since among

the six tuples that satisfy p, two are misclassified.

Our goal is to detect significant groups (that is, large enough) that are treated unfairly by the algorithm (that is, have low accuracy compared with the overall algorithm accuracy). In particular, we wish to provide the user with a concise description of these groups. Given a classifier M and an error threshold τ_a, we say that p is a *most general* pattern with accuracy below τ_a if $acc_M(p) \leqslant \tau_a$ and $\forall p' \subsetneq p$, $acc_M(p') \geqslant \tau_a$. Our problem is then, given a database D, a trained classifier M with overall accuracy a, an accuracy delta threshold $\Delta\tau_a$, and a size threshold τ_s, find all most general patterns with size $\geqslant \tau_s$ and accuracy $\leqslant \tau_a$, where $\tau_a = a - \Delta\tau_a$. We next outline our solution.

Algorithm. Given a labeled test dataset D, a classifier M, and thresholds τ_s and $\Delta\tau_a$ over the size and accuracy of the patterns respectively, we first use M and D to compute the set of misclassified tuples D^M_{mis} (the tuples in D classified incorrectly by M) and $\tau_a = a - \Delta\tau_a$. We then traverse the set of possible patterns (starting with the most general ones) and compute the accuracy for each pattern. To do so, we use the notion of *pattern graph* presented in Asudeh et al.[5] Briefly, the nodes in the graph are the set of all possible patterns, and there is an edge between a pair of patterns p and p' if $p \subset p'$ and p' can be obtained from p by adding a single attribute value pair. In this case, we say that $p(p')$ is a parent (child) of $p'(p)$. As shown in Asudeh et al.,[5] the pattern graph can be traversed in a top-down fashion, while generating each pattern at most once. Moreover, the size of a pattern p in a dataset can be computed from its parents. We build upon this observation to compute the size of each pattern p in the given dataset D and in the misclassified dataset D^M_{mis} and then use them to compute the pattern accuracy $acc_M(p)$.

Pruning. To optimize the search, we prune the search space. First note that $c_D(p) \geqslant c_D(p')$ for every p and p' when p' is a descendent of p in the pattern graph. Thus, when reaching a pattern p with $c_D(p) < \tau_s$, the descendent of p can be pruned. Moreover, if $c_D(p) \geqslant \tau_s$ and $acc_M(p) \leqslant \tau_a$, its descendent can be pruned as well since we are looking for most general patterns. Note: the accuracy of a pattern p is lower than τ_a if

$$acc_M(p) = 1 - \frac{c_{D^M_{mis}}(p)}{c_D(p)} \leqslant \tau_a,$$

that is, $c_D(p)\,(1-\tau_a) \leqslant c_{D^M_{mis}}(p)$.

The challenge for a data-science pipeline is that it has to establish end-to-end reliability.

Since we are interested only in patterns p with $c_D(p) \geqslant \tau_s$ we get $\tau_s \cdot (1-\tau_a) \leqslant c_{D^M_{mis}}(p)$. Thus, patterns with size less than $\tau_s \cdot (1-\tau_a)$ can be pruned as well.

Example 9. Consider again the dataset in Figure 2 and the classifier M whose results are depicted in the Prediction column. Given the thresholds $\tau_s = 5$ and $\Delta\tau_a = 0.2$, the algorithm first computes τ_a and the set of misclassified tuples D^M_{mis}. In this case, the overall accuracy of the classifier is 0.875, thus $\tau_a = 0.675$ and $D^M_{mis} = \{3, 13\}$. The most general patterns are $p_1 = \{\text{Gender} = \text{M}\}$, $p_2 = \{\text{Gender} = \text{F}\}$, $p_3 = \{\text{School} = \text{MS}\}$, $p_4 = \{\text{School} = \text{GP}\}$, $p_5 = \{\text{Address} = \text{U}\}$, $p_6 = \{\text{Address} = \text{R}\}$, $p_7 = \{\text{Failures} = 0\}$, $p_8 = \{\text{Failures} = 1\}$, and $p_9 = \{\text{Failures} = 2\}$. The algorithm first considers p_1. Since $c_{D^M_{mis}}(p_1) = 1 < (1-\tau_a) \cdot \tau_s = 1.625$ this pattern and all of its descendants can be pruned. Similarly, p_2 and p_3 and their descendants are pruned. Next, the algorithm considers p_4. There are two students in the GP school with incorrect predicted grades, thus $c_{D^M_{mis}}(p_4) = 2 > 1.625$. Since the total number of students in GP is 8, the accuracy of p_4 is $acc_M(p_4) = 0.75 > \tau_a = 0.675$, thus its children are generated. Similarly, the children of p_5 are generated. Patterns p_6 through p_9 (and their descendants) are pruned as well. The patterns p_6 and p_9 are pruned due to their low number of misclassified tuples. For the patterns p_7 and p_8, $c_D(p_7) = c_D(p_8) = 4 < \tau_s = 5$, thus they and their descendants are pruned. Finally, the pattern $p_{10} = \{\text{School} = \text{GP}, \text{Address} = \text{U}\}$ (generated as a child of p_4) is considered. Since $c_D(p_{10}) = 6$ and $c_{D^M_{mis}}(p_{10}) = 2$ the accuracy of p_{10} is 0.67 < 0.675 and p_{10} is added to the result set. In this example, this is the only pattern with low accuracy.

Generalized solution. We now propose a generalization of the algorithm described in the section "Detecting groups with low accuracy" to account for other definitions for algorithmic fairness.[21] In particular, we consider

Figure 4. Aggregation and refinement query results.

Address	AVG_Grade
R	11.75
U	10.38

(a) Result of query Q

Address	Failures	AVG_Grade
R	0	12
	1	10
	2	9
U	0	19
	1	7.5
	2	12

(b) Result of refinement query Q'

Figure 5. The set of possible refinement queries, their support, and weight.

weight	WHERE	GROUP BY	
		Address	Address, Failures
1	–	1	$\frac{7}{9}$
0.5	Gender = M	1	1
0.5	Gender = F	0	$\frac{1}{3}$
0.5	School = GP	0	0
0.5	School = MS	1	1
0.25	Gender = MandSchool = GP	0	$\frac{1}{3}$
0.25	Gender = MandSchool = MS	0	0
0.25	Gender = FandSchool = GP	0	$\frac{1}{3}$
0.25	Gender = FandSchool = MS	1	1

definitions based on the model's prediction and actual outcomes. These statistical measures of fairness rely on the possible cases of a classifier's outcome: True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). We next offer a brief overview of such definitions[d] and refer readers to Verma and Rubin[21] for more details.

Predictive parity. The fraction of correct positive prediction $\frac{TP}{TP+FP}$ should be similar for all groups.

False-positive error-rate balance (predictive equality). The probability of a subject in the actual negative class to have a positive predictive value $FPR = \frac{FP}{FP+TN}$ is similar for all groups.

False-negative error-rate balance (equal opportunity). Similar to the above, but considers the probability of falsely classifying subjects in the positive class as negative $FNR = \frac{FN}{TP+FN}$.

Equalized odds. Combines the previous two definitions. All groups should have both similar false-positive error-rate balance $\frac{FP}{FP+TN}$ and false-negative error-rate balance $\frac{FN}{TP+FN}$.

Conditional use accuracy equality. All groups should have similar probability of subjects to be accurately predicted as positive $\frac{TP}{TP+FP}$ and accurately predicted as negative $\frac{TN}{TN+FN}$.

Treatment equality. This considers the ratio of error. A classifier satisfies it if all groups have a similar ratio of false negatives and false positives.

Generalized algorithm. Our solution can be generalized to capture these fairness definitions (and any definition that relies on the values TP, FP, FN, and TN). The general algorithm is similar to the algorithm for detecting patterns with low accuracy with the following modifications. First, the algorithm computes four datasets (instead of D^M_{mis}): D^M_{TP}, D^M_{FP}, D^M_{FN}, and D^M_{TN} containing the tuples from D that are TP, FP, FN, and TN, respectively. The algorithm traverses the pattern graph in a top-down fashion. For each pattern, it computes the size, and the fairness measure using D^M_{TP}, D^M_{FP}, D^M_{FN}, and D^M_{TN}. We use τ_f to refer the fairness measure in the general case. Note that for some fairness measures (for instance, false positive/negative error-rate balance) lower values are preferred. In this case, the algorithm returns groups with a fairness value higher than τ_f. Moreover, note that the pruning based on $c_{D^M_{mis}}$ as presented is not applicable in general; however, other methods, such as pruning based on the numerator value, can be used.

The proposed method aims to assist users in assessing the fairness of an ML model for any given test data. This is in line with the common practice of performance analysis of ML models. Consequently, a limited amount of data in the test set may affect the results. In particular, the test data should be representative.

Reliable Result Derivation

In the process of data analysis and the development of data-driven tools, query results are often conveyed and even used in practice, based on *generalizations* that represent "the main takeaway" from the analysis. In this section, we describe our model for evaluating the quality of generalizations derived by aggregate queries.

Bias in the result of aggregation queries was studied—for example, in Salimi et al.,[18] which use causal analysis to define the bias. As such, they focus on inferring covariates, attributes that are correlated with the aggregation result, whose distribution differs in different groups. Our model aims at assessing the quality of the results by examining the subgroups of the groups in the output. An interesting application of our proposed model is the detection of the Simpson's paradox that was also studied in Guo et al.[10] We start by presenting the notion of statements based on aggregation queries and then discuss their score, which reflects the degree to which the result of aggregation represents the underlying data.

Aggregation query-based statements and refinement queries. We consider statements on the relation between groups in the results of an aggregation query.

Example 10. Consider the Grades table in Figure 2 and the following aggregation query Q:

```
SELECT Address, avg(Grade)
as AVG_Grade
FROM Grades
GROUP BY Address
```

The result of the query is shown in

d The definitions given in Verma and Rubin[21] consider two groups (protected/unprotected groups defined using a sensitive attribute). We generalize the definitions to fit our problem of detecting unfairly treated groups.

Figure 4a. Based on the result, we can state that (*S*): on average, the grades of students with rural addresses are higher than the grades of students with urban addresses.

To explore the possible refined subgroup entities, we next define the notion of *query refinement*. Intuitively, the groups in the results of an aggregation query Q can be refined either by adding attributes to the WHERE clause of Q (that is, adding attribute-value assignments), or the GROUP BY clause (that is, adding grouping attributes).

Example 11. Consider again the query Q from Example 10. A possible refinement query Q' of Q is

```
SELECT Address, Failures,
avg(Grade) as AVG_Grade
FROM Grades
WHERE Gender = F
GROUP BY Address, Failures
```

In this example, Q was refined by adding Gender = F to the WHERE clause and adding Failures to the GROUP BY clause. The result of the refined query is given in Figure 4b. Adding an attribute to the WHERE or the GROUP BY clause allows for the comparison of the aggregation results of subgroups, which can determine how well the underlying data is reflected by the query result and reveal misleading conclusions. For instance, the statement *S* from Example 10 is supported by the result of the given query Q but does not reflect the fact that this is not th case if we consider only female students, for which the grades of students from urban addresses are higher.

Statement's score. Given a statement and the set of refinement attributes (to be added to the WHERE and GROUP BY clause), our goal is to define the score of the statement, which measures how well it reflects the underlying data. We do it by considering all possible refinements of the given query Q and taking into account the size of the subgroups in the result as their potential influence on the score. To this end, we define the *weight of a refinement query* and the *support of a statement*. Using these two notions, we define the score of a statement.

The weight of a refinement query. Given a set of attributes with value assignments $A_i = a_i$ (not appearing in the original query) such that A_i is an attribute and a_i is a value in the domain of A_i, the weight of a refinement query is the relative size of the tuples that confirm the value assignment from the total size of tuples involved in the statement groups.

Once the ML model is trained, data analysts may wish to validate its fairness toward different groups in the data.

Example 12. Consider the query Q' that refine the query Q given in Examples 11 and 10 respectively. The statement *S*, based on the original query Q, considers students from rural and urban addresses. There are 16 such tuples in the data (all tuples in this case). Eight of those tuples confirm the condition Gender = Female; namely, the relative portion of female students is $\frac{8}{16} = \frac{1}{2}$, and so is the weight of Q'.

The support of a statement. To measure the support of a statement, we consider pairwise comparison between subgroups of the statement's groups, obtained by refinement of the GROUP BY clause. The support is then the number of subgroup comparisons that align with the original statement divided by the total number of comparisons applied.

Example 13. The refinement query Q' refines each group in the statement *S*: students with a rural address and an urban address, into three subgroups based on the number of past failures. To compute the support of Q' in *S*, we apply pairwise comparison between the subgroups—that is, the average grade of students with a rural address and 0, 1, and 2 past failures with the average grade of students with a rural address and 0, 1, and 2 past failures. The total number of comparisons applied in this example is nine, out of which there were only three cases where the average grade of students with a rural address is higher than the grade of students with an urban address (when comparing the grades of students from rural addresses with 0, 1, and 2 failures to those with urban addresses and a single past failure). Thus, the support of the query Q' is $\frac{3}{9} = \frac{1}{3}$.

The score of a statement. Finally, the score of a statement is defined with respect to a set of refinement attributes $\mathcal{A}_{pred}$ and $\mathcal{A}_{grp}$, which define the attribute that can be used to

refine the aggregation query through the WHERE and GROUP BY clauses respectively. Given a statement S_Q based on the result of a query Q and the sets of refinement attributes $\mathcal{A}_{pred}$ and $\mathcal{A}_{grp}$, let $\mathcal{R}(Q)$ be the set of all possible refinement queries of Q obtained using $\mathcal{A}_{pred}$ and $\mathcal{A}_{grp}$. The score of S_Q is then defined as the sum of the product of the weight and score for every possible refinement query, normalized by the sum of weights of all refinement queries.

$$score(S_Q) = \frac{\sum_{Q' \in \mathcal{R}(Q)} weight(Q') \cdot supp(Q')}{\sum_{Q' \in \mathcal{R}(Q)} weight(Q')}$$

Example 14. Consider again the query Q and the statement S given in Example 10, and let $\mathcal{A}_{pred} = \{\texttt{Gender}, \texttt{School}\}$ and $\mathcal{A}_{grp} = \{\texttt{Failures}\}$. The set of all possible refinement queries is summarized in Figure 5. Rows correspond to refinements through the WHERE clause, and columns to refinements through the GROUP BY clause. The value in each cell depicts the support of the corresponding refinement query. For instance, the value $\frac{1}{3}$ in the second column of the third row is the support of the query Q' given in Example 11. The weights of the queries are shown next to each row (recall that the weight is determined merely by the WHERE clause; thus, queries in the same row have equivalent weight). There are 18 refinement queries with total weight of 8, and the sum of product of the weight and support for every possible refinement query is 4.61; thus, the score of the statement S is $\frac{4.61}{8} = 0.576$.

Intuitively, the score reflects the fraction of subgroup populations supporting the generalization. A higher score (close to 1) indicates a better reflection of the underlying data. The score of the statement S in the running example is relatively low, although it is supported by the result of the query Q. The reason is that there is a large fraction of subgroups, such as female students or students from the GP school opposing the statement.

Refinement attributes. The attribute sets $\mathcal{A}_{pred}$ and $\mathcal{A}_{grp}$ can be defined by the end user; however, we propose a default setting. Given a query Q, attributes that share (almost) the same data domain in all groups of the result of applying Q are added to $\mathcal{A}_{pred}$ (used for "similar" subgroup comparison). Attributes having unique values (for example, ID, names) are ignored. The rest are added to $\mathcal{A}_{grp}$.

Score computation. A naive algorithm for score computation would iterate over the set of all possible refinement queries, compute the weight and support for each query, and then use them to compute the score. The number of refinement queries is exponential in $|\mathcal{A}_{pred}|$ and $|\mathcal{A}_{grp}|$. Executing all queries for weight and support computation may lead to prohibitive execution time. Lin et al.,[13] presents an improved algorithm that works well in practice (as we show in our experimental study). The algorithm uses a Hasse diagram, which represents a partial order over the refinement queries, based on their WHERE and GROUP BY attributes. Enumerating the refinement queries in a bottom-up fashion allows for the reuse of computational results (using a dedicated data structure) and thus refrains from repeated access to the dataset to get aggregate results. This significantly reduces running time.

Given a low score, the user may wish to: (i) understand why the score is low (that is, which parts of the data do not "agree with the statement") and (ii) refine the statement, such that the new refined statement better represents the data but is "as close as possible" to the original query. To provide the user with a better understanding of the resulting score, Lin et al.[13] introduces the problem of providing *counterexamples*—that is, disclosing significant parts of the data opposing the statement. We further discuss the problem of refining the statement to obtain more representative alternatives for identifying counterexamples and statement refinement using an inverted index structure (see Lin et al.[13]).

Conclusion

We have presented data-centric methods designed to increase the reliability of data-driven decision systems. While limited to categorical data, these methods may be used at different points in developing the data-driven decision-system pipeline. We believe that multiple such tools will need to be used together to be able to construct data-based decision systems with end-to-end reliability.

Acknowledgments

This research has been supported in part by NSF grants 1741022, 1934565, and 2106176.

References

1. AI Fairness 360; https://aif360.mybluemix.net/.
2. Abedjan, Z., Golab, L., and Naumann, F. Profiling relational data: A survey. *Intern. J. of Very Large Databases 24*, 4 (2015), 557–581.
3. Agrawal, R. and Srikant, R. Fast algorithms for mining association rules in large databases. In *Proceedings of the Intern. Conf. on Very Large Databases* (1994), Morgan Kaufmann, 487–499.
4. Angwin, J. et al. Machine bias. *ProPublica* (2016).
5. Asudeh, A., Jin, Z., and Jagadish, H.V. Assessing and remedying coverage for a given dataset. *IEEE 35th Intern. Conf. on Data Engineering* (2019), 554–565.
6. Baralis, E., Paraboschi, S., and Teniente, E. Materialized views selection in a multidimensional database. In *Proceedings of the 23rd Intern. Conf. on Very Large Databases* (1997), Morgan Kaufmann, 156–165.
7. Broussard, M. When algorithms give real students imaginary grades. *The New York Times* (2020); https://www.nytimes.com/2020/09/08/opinion/international-baccalaureate-algorithm-grades.html
8. Cortez, P. and Silva, A.M.G. Using data mining to predict secondary school student performance. In *Proceedings of the 5th Annual Future Business Tech. Conf.* (2008).
9. Deshpande, A., Garofalakis, M.N., and Rastogi, R. Independence is good: Dependency-based histogram synopses for high-dimensional data. *ACM SIGMOD* (2001), 199–210.
10. Guo, Y., Binnig, C., and Kraska, T. What you see is not what you get! Detecting Simpson's Paradoxes during data exploration. In *Proceedings of the 2nd Workshop on Human in the Loop Analytics* (2017), 2:1–2:5.
11. Jo, S. et al. Verifying text summaries of relational data sets. *ACM SIGMOD* (2019), 299–316.
12. Li, J., Moskovitch, Y., and Jagadish, H.V. Denouncer: Detection of unfairness in classifiers. In *Proceedings of the VLDB Endowment 14*, 12 (2021), 2719–2722.
13. Lin, Y. et al. On detecting cherry-picked generalizations. In *Proceedings of the VLDB Endowment 15*, 1 (2021), 59–71.
14. Moskovitch, Y. and Jagadish, H.V. COUNTATA: Dataset labeling using pattern counts. In *Proceedings of the VLDB Endowment 13*, 12 (2020).
15. Moskovitch, Y. and Jagadish, H.V. Patterns count-based labels for datasets. *IEEE 35th Intern. Conf. on Data Engineering* (2019), 1961–1966.
16. Müller, M., Moerkotte, G., and Kolb, O. Improved selectivity estimation by combining knowledge from sampling and synopses. In *Proceedings of the VLDB Endowment 11*, 9 (2018), 1016–1028.
17. Rymon, R. Search through systematic set enumeration. *Intern. Conf. on Principles of Knowledge Representation and Reasoning* (1992), Morgan Kaufmann, 539–550.
18. Salimi, B., Gehrke, J., and Suciu, D. Bias in OLAP queries: Detection, explanation, and removal. *ACM SIGMOD* (2018), 1021–1035.
19. Sissons, B. What to know about ADHD misdiagnosis. *Medical News Today* (2019); https://www.medicalnewstoday.com/articles/325595#age-related-factors.
20. Stoyanovich, J. and Howe, B. Nutritional labels for data and models. *IEEE Data Engineering Bulletin 42*, 3 (2019), 13–23.
21. Verma, S. and Rubin, J. Fairness definitions explained. *IEEE/ACM FairWare* (2018), 1–7.
22. Yang, Z. et al. Deep unsupervised cardinality estimation. In *Proceedings of the VLDB Endowment 13*, 3 (2019), 279–292.

Yuval Moskovitch (yuvalmos@bgu.ac.il), assistant professor in the Computer Science dept. at Ben-Gurion University of the Negev, Beersheba, Israel, worked on this article as a postdoctoral research fellow in the Electrical Engineering and Computer Science dept. at the University of Michigan, Ann Arbor, MI, USA.

H.V. Jagadish is Edgar F. Codd Distinguished University Professor and Bernard A. Galler Collegiate Professor of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA.