Discovering Unwarranted Associations in Data-Driven Applications with the FairTest Testing Toolkit

Florian Tramèr[1], Vaggelis Atlidakis[2], Roxana Geambasu[2], Daniel Hsu[2],
Jean-Pierre Hubaux[1], Mathias Humbert[4], Ari Juels[3], and Huang Lin[1]
[1]*École Polytechnique Fédérale de Lausanne* — [2]*Columbia University* — [3]*Cornell Tech* — [4]*Saarland University*

Abstract

In today's data-driven world, programmers routinely incorporate user data into complex algorithms, heuristics, and application pipelines. While often beneficial, this practice can have unintended and detrimental consequences, such as the discriminatory effects identified in Staples' online pricing algorithm and the racially offensive labels recently found in Google's image tagger.

We argue that such effects are *bugs* that should be tested for and debugged in a manner similar to functionality, performance, and security bugs. We describe *FairTest*, a testing toolkit that detects *unwarranted associations* between an algorithm's outputs (*e.g.,* prices or labels) and user subpopulations, including protected groups (*e.g.,* defined by race or gender). FairTest reports any statistically significant associations to programmers as potential bugs, ranked by their strength and likelihood of being unintentional, rather than necessary effects.

We designed FairTest for ease of use by programmers and integrated it into the evaluation framework of SciPy, a popular library for data analytics. We used FairTest experimentally to identify unfair disparate impact, offensive labeling, and disparate rates of algorithmic error in six applications and datasets. As examples, our results reveal subtle biases against older populations in the distribution of error in a real predictive health application, and offensive racial labeling in an image tagger.

1 Introduction

Today's applications – ranging from simple mobile games to complex web applications – are increasingly *data-driven*. User data, such as clicks, locations, and social information, can enhance user experience by letting applications customize their functionality, contents, and offers according to individual preferences. Data can also improve business revenues by enabling effective product placement and targeted advertising. Finally, incorporating data into a wide spectrum of societal processes, including healthcare, crime prevention, and emergency response, promises to vastly improve their efficiency.

Despite these benefits, integrating user data into applications and decision making can have unintended and detrimental consequences that are often difficult to anticipate. A case in point is the Staples differential pricing case [52]. Staples' seemingly rational decision to adjust online prices based on user-proximity to competitor stores led to consistently higher prices for low-income customers, who (as it turns out) generally live farther from these stores. Staples' intentions aside, the difficulty of foreseeing all subtle implications and risks of data-driven heuristics is clear. And the risk of such unintended side-effects will only increase as new kinds of personal and user-generated data – *e.g.,* collected through the Internet of Things – are passed through increasingly complex machine learning algorithms, whose associations and inferences are (arguably) impossible to foresee.

It is no wonder, then, that reports of discriminatory effects in data-driven applications litter the news. Google's image tagger was recently found to associate racially offensive labels with images of black people [19]. Discriminatory online advertising has been shown to associate ads for lower-paying jobs with women [8] and offensive, racially charged ads with black people [49].

We argue that such algorithmic biases – which we generically call *unwarranted associations* – are new kinds of *bugs* specific to modern, data-driven applications, which programmers should actively test for, debug, and fix with the same urgency as they apply to functionality, performance, and security bugs. Unwarranted association bugs may offend or even harm users. They can cause programmers and businesses embarrassment, mistrust, and potentially loss of revenue. Finally, unwarranted associations may be symptoms of an actual malfunction of a data-driven algorithm, such as a machine learning algorithm exhibiting poor accuracy for minority groups that are under-represented in its training set [23].

We present *FairTest*, a testing toolkit for data-driven applications that helps programmers test for and to some extent debug unwarranted associations. At its core, FairTest detects statistically significant associations between an algorithm's outputs (*e.g.,* prices or labels) and user subpopulations, including protected groups (*e.g.,* those defined by race, gender, or income level). It then reports any statistically significant effects as *association bugs*, filtered and ranked by their strength, statistical significance, and likelihood of being unintended side-effects rather than necessary consequences of an application requirement. FairTest identifies both weak associations that affect large populations and strong associations that affect smaller subpopulations. For example, our simulation of Staples' pricing scheme fed with data from the U.S. census revealed that while some disparate impact on low-income populations arises across the entire U.S., certain parts of the country, such as New York state, exhibit stronger discriminatory effects.

A key innovation in FairTest is its *guided decision tree construction* algorithm, which efficiently derives semantically meaningful subpopulations that appear most affected by an association bug. In prior work, detecting such subpopulations has required uninformed, exhaustive searches for hidden associations [38, 47, 48]. Inspired by decision tree algorithms from machine learning, our algorithm recursively splits the user population into subsets so as to maximize some association metric between algorithm outputs (*e.g.,* a price) and protected user attributes (*e.g.,* race). At every step, the tree therefore discovers subpopulations of decreasing size but with increasingly stronger discriminatory effects.

This core functionality is surprisingly flexible, enabling a wide variety of investigations that programmers may wish to perform on their data-driven applications. At present, FairTest supports three main investigation types: (1) *Discovery* of potential association bugs with limited a priori knowledge of what bugs an application may present, (2) *Testing* for one or a few suspected association bugs (*e.g.,* higher prices or denied loans), and (3) *Error profiling* of a machine learning (ML) algorithm over a user population, that is, identifying any subpopulations with which erroneous predictions are disparately associated. In conjunction with these core investigation types, FairTest can assist in preliminary *debugging* of uncovered association bugs, by allowing a developer to rule out potential confounders for observed unfair effects.

We used FairTest to test for disparate impact, discover offensive labeling, and profile algorithmic errors in six applications and datasets, including: a simulation of Staples' pricing scheme fed by U.S. census data [50], a movie recommender, a predictive health application, and an image tagger. We found association bugs in all cases, demonstrating the critical need for tools like FairTest.

Overall we bring the following contributions:

1. We introduce *unwarranted associations*, a type of bug specific to emerging data-driven applications. While prior works have raised the broad concern of unintended, unfair consequences of algorithmic decision making, our definition is more general and broadly applicable than previous fairness definitions.
2. We design, implement, and evaluate *FairTest*, the first testing tool for unwarranted associations. It uniquely supports: (1) multiple association metrics as required by various applications and use cases, (2) efficient detection of association bugs in user subpopulations, (3) rigorous statistical result assessments, and (4) to some extent debugging of discovered associations.
3. We develop a *guided decision tree construction* algorithm, which "zooms into" a user population to find meaningful subpopulations strongly affected by a bug.
4. We integrate FairTest into SciPy in support of association bug discovery, testing, and error profiling.
5. We demonstrate FairTest's three investigation types on six real-world applications and datasets, revealing the widespread occurrence of association bugs, as well as FairTest's effectiveness in detecting them.
6. We will release FairTest's source code on publication.

2 Motivation and Goals

Our research aims to: (1) demonstrate the importance of testing for unwarranted associations in data-driven applications, and (2) develop tools to assist programmers in finding and investigating such bugs.

2.1 Motivating Examples

Typical examples of unwarranted associations in the related literature focus on high-stakes processes where differential treatment or impact is punishable by law, *e.g.,* hiring, providing credit, or offering housing. While such sensitive applications indeed require close inspection, we argue that *any* application that ingests and processes user data deserves scrutiny for association bugs. We present three examples that underscore the diverse contexts in which unwarranted associations can arise, and illustrate the capabilities needed to detect them:

- *Google Photos.* Google's recently released Photos application includes an ML-based image tagging system. Users found that Photos produced offensive labels, tagging black people in photos as "gorillas" [19]. Google promptly apologized for the bug, saying that "This is 100% not OK," and promised to fix it [19].
- *Staples' differential pricing scheme.* The office retailer Staples implemented what seemed a rational differential pricing scheme for online purchases: users located within approximately 20 miles of a rival store (*e.g.,* Office Depot) were often offered a discounted price. Investigators at the Wall Street Journal found that the pricing scheme had a negative disparate impact on low-income customers [52]. The investigators called the situation an "unintended side-effect" [52].
- *Healthcare prediction.* Based on real-world data and a winning approach from the Heritage Health Prize Competition [25], we built a model using past healthcare claims to predict a user's number of hospital visits in the next year. Using FairTest, we found that although the model is highly accurate overall, its errors are unevenly concentrated on elderly users, especially in subpopulations with certain pre-existing conditions. If an insurance company used this algorithm to tune insurance premiums, they might involuntarily discriminate against elderly people within these populations.

These examples illustrate the wide variety of unwarranted associations and the importance of proactively testing for and remedying them before they can harm users or embarrass companies. These settings also illustrate the major capabilities that are needed to address these concerns. Google's Photos case shows that unwarranted associations can be difficult to anticipate, and therefore that tools are required for *discovery* of such bugs. Staples' mishap illustrates the need, given a possible or suspected discriminatory effect, for tools that enable rigorous *testing* of its presence, extent, and impact. Finally, our healthcare prediction example shows that rather than yielding outcomes that are discriminatory in their content, an application can disparately impact certain subpopulations in the form of uneven error rates. This highlights the need for principled *error profiling* of ML algorithms and dependent applications.

Our goal, and insight, in this paper is to develop a comprehensive system design, together with the necessary definitional support, to enable investigations of all of these different types of unwarranted associations.

2.2 Candidate Approaches and Related Work

A number of intuitive approaches may appear applicable to discover, debug, or prevent association bugs. Upon closer examination, we find none of these to be sufficient. First, a developer might explicitly omit protected user attributes, such as gender or race, from an application's inputs in an attempt to avoid discrimination along these axes. Unfortunately, this is insufficient, as subtle associations between protected attributes and actual program inputs (e.g., locations, preferences) may result in indirect biases. For example, Staples' algorithm did not explicitly ingest information about the socio-economic status of its customers; yet location surfaced as an unanticipated *proxy* for this sensitive information.

Second, a programmer might inspect discriminatory effects only at a coarse-grained level, *i.e.,* over a global population of application users. For instance, she could check whether her pricing algorithm gives similar prices, on average, to low-income and high-income users across the U.S. However, disparities observed over a population may differ from, or even contradict, those found in smaller subsets, an effect known as *Simpson's paradox*. The 1973 Berkeley admissions [2] are a famous example: admission rates appeared to disfavor women, yet individual departments exhibited either no bias or a reverse bias. Incidentally, the adverse effects of Staples' pricing scheme also varied significantly from state to state [52].

Third, a programmer might look for disparate effects in a few subpopulations of interest (*e.g.,* due to historical relevance). This appears to be a common practice among discrimination watchdogs. Unfortunately, this approach is neither systematic nor scalable: even for datasets with a few dimensions, the number of possible subpopulations can be impractically large, and manual inspection of a few may miss important effects.

Fourth, debugging the ML component in an application is often conducted in an *ad hoc* fashion, with the ML practitioner manually employing exploratory data analysis to understand data features and modeling errors [24, 29, 40, 41]. Applying these practices to our use cases is insufficient. A challenge that remains is the systematic inspection of many combinations of features that may define meaningful subpopulations. Existing automated tools for deriving subpopulations (*e.g.,* clustering algorithms [24]) are not guided by the measures that define the potential discriminatory effects, and do not always produce interpretable subpopulations. Although some learning methods allow post-learning introspection (*e.g.,* Random Forests [5]), not all applications use such methods, and hence an algorithm-independent method is desirable.

Finally, the emerging "algorithmic fairness" field aims to develop methods to detect or avoid discrimination in data-driven applications. Our study of this literature reveals some serious limitations:

At a foundational level, we find a proliferation of fairness definitions, each with at least one of three major drawbacks: (1) They apply only to simple binary (protected) attributes, *e.g.,* favored and disfavored status [7, 16, 20, 30, 31, 38, 42, 47, 48, 56, 57], or are instead difficult to test using a sample [12]; (2) They ignore disparities in user subpopulations [7, 16, 30–32, 57]; (3) They fail to consider reasonable explanations that may negate the significance of an association [7, 16, 30–32, 38, 42] (*e.g.,* a movie recommender may suggest poor-quality movies to a subgroup, but that may be explained by the group members' preferences).

At a system design level, the fragmentation of incomplete fairness definitions leads to designs with limited applicability and low prospects for real-world adoption. Indeed, most papers in this space report experimentation on just one or two small datasets [7, 12, 20, 30–32, 47, 48, 57]. No exploration exists for challenging systems issues, such as building a generic system that supports many applications and investigation types, ensuring ease of use for programmers, allowing debugging of detected associations, or scaling to large datasets.

Detailed discussion of related work is in Appendix B.

2.3 Design Goals

We designed the FairTest testing toolkit for unwarranted associations informed by the preceding examples and limitations of prior approaches. Its primary goal is to *detect* unwarranted associations by supporting three core types of investigations: (1) ***Testing*** for one or a few suspected association bugs (e.g., higher prices or denied loans for certain populations), in order to discover any subpopulations disparately impacted by these potential bugs; (2) ***Discovery*** of potential association bugs; that is, identifying unanticipated, questionable associations (such as labels associated with particular subpopulations) with little a priori knowledge of what bugs an application may present or what subpopulations these bugs may affect; and (3) ***Error profiling*** of a ML algorithm over a user population, that is, identifying any subpopulations with which erroneous algorithmic outputs are disparately associated; while a form of testing, error profiling treats not application outputs, but application accuracy.

FairTest must meet the following requirements:

- *Generic and broadly applicable:* FairTest's design, as well as the definitional foundations it relies upon, must be generic and broadly applicable.
- *Ease of use:* FairTest must not assume that data-driven programmers are expert statisticians. It must therefore: (1) provide complete and directly interpretable information for every association bug, including rigorous measures of statistical significance and of effect size, and (2) filter and rank bugs by their "importance" to help programmers prioritize their efforts.
- *Detect unwarranted associations of varied importance:* The "importance" of an association bug may be measured either by the size of the population it affects (*e.g.,* many people may get poor movie recommendations) or by the strength of its effect (regardless of affected population size). FairTest must discover both systematic discrimination over large populations and severe disparities exhibited in specific subpopulations.
- *Some support for debugging:* In addition to primitives for association bug detection, FairTest aims to provide some basic, if incomplete, primitives to help programmers narrow down the root cause of these bugs.

2.4 Threat Model and Assumptions

FairTest is designed to aid developers in discovering association bugs. It is thus intended to be used by honest developers to study honestly designed applications. Applications do not intentionally induce unwarranted associations or seek to conceal associations from FairTest.

FairTest's debugging is restricted to identifying any *confounding* factors that might explain an association bug (*e.g.,* in the Berkeley admissions [2], the department to which a person applied to was a confounder). While this is valuable, confounders are not the only cause of association bugs. Other potential causes, such as insufficient training data for an ML algorithm, are out of scope.

While FairTest provides abstractions for association bug *detection* and to some extent *debugging*, it offers no explicit support for *remediation*. Indeed, traditional testing tools (e.g., for functionality or performance bugs) rarely prescribe fixes for the bugs they reveal; developers use them to find and understand bugs, and then develop their own fixes, which can range from trivial changes to major application restructuring. In practice, however, with FairTest, as with other testing tools, fixes often become apparent once a developer understands a bug. We show throughout the paper, and in §6.3 in particular, cases where debugging yields clear remediation paths.

A key assumption in FairTest is that a developer will have at her disposal a set of protected attributes (*e.g.,* gender, race, income level) for her users. For some of these attributes, it is plausible that programmers have this information from user profiles. In other cases, public datasets, such as the U.S. census data, can be leveraged to test for unwarranted associations on attributes that the programmer lacks (see §3.1). Finally, §5 discusses a deployment model that relaxes the above assumption: an architecture where a trusted auditor (*e.g.,* EFF) collects protected attributes from a large user population and runs FairTest on programmers' behalf. In this case, the auditor is fully trusted by programmers and users.

Regardless of the data's source, we assume that it is *representative* of an application's user population and not tainted by selection bias. Moreover, while we aim to discover associations in smaller subpopulations, we do not aim to discover tiny-scale associations (*e.g.,* at the level of an individual). An application may behave poorly for a specific user, yet FairTest will not detect that.

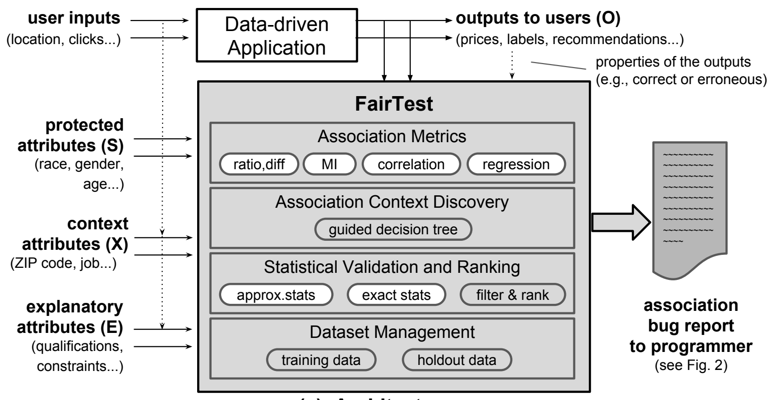
3 FairTest Overview

Fig.1(a) shows the FairTest architecture. At a high level, the data-driven application – the object of FairTest's investigations – takes inputs from each user, such as locations or clicks, and returns a set of outputs to the user. To run an investigation, the developer supplies FairTest with a dataset consisting of a number of attributes from application users, along with the outputs (or properties of the outputs) for those users. FairTest analyzes this data and returns an *association report*. The report lists statistically significant associations that FairTest has found between specified protected attributes $\boldsymbol{S}$ (such as race or gender) and the outputs $\boldsymbol{O}$. The programmer then inspects the report and determines which reported associations are real bugs that require fixing and which are admissible effects in the context of her company's policies. After giving a concrete example of an association report, we detail FairTest's architecture and algorithm in the remainder of this section. §4 then details our design.

3.1 Association Report Example

Suppose that Staples' programmers wished to inspect their pricing scheme's impact on the users before deploying it in production (e.g., out of principle or to avoid bad publicity). To do so, they could use U.S. census statistics [50] to emulate users with realistic demographics visiting their website from various locations. They would run their location-based pricing scheme for those users and use FairTest's *Testing investigation* to test for disparate impact on race, income, or other sensitive groups.

We ran such an investigation on a simulated pricing scheme akin to Staples', which gives discounts to users



(a) Architecture

```
Input: Data D = (S, X, E, O);
Output: Association bug report.
1. Split D into D_train and D_test.
2. for each protected attribute S_i in S:
   2.1. Using D_train, derive association contexts
        by building a decision tree on X guided
        by the value of association metric M
        between S_i and O.
   2.2. for each context:
        Using D_test, compute confidence
        interval (CI) and statistical significance
        (p-value) for M in this context.
3. Correct CIs, p-values for multiple testing
   across all protected attributes and contexts.
4. for each protected attribute S_i in S:
   4.1. Filter results on p-value.
   4.2. Rank results on CIs.
5. return association bugs for each S_i.
```

(b) Basic Algorithm

Fig. 1: **FairTest Architecture and Algorithm.** (a) Grey boxes denote FairTest components. Rounded boxes denote specific mechanisms. White rounded boxes denote extensibility points; transparent rounded boxes denote core, generic mechanisms in FairTest. (b) $\boldsymbol{S}$, $\boldsymbol{X}$, $\boldsymbol{E}$ denote protected, context, and explanatory attributes, respectively; $\boldsymbol{O}$ denotes outputs. See (a) for examples of these attributes.

```
Report of associations of O=Price on Si=Income:
Association metric: norm. mutual information (NMI).
Global Population of size 494,436
p-value = 3.34e-10 ; NMI = [0.0001, 0.0005]
Price | Income <$50K | Income >=$50K |        Total
High  |   15301 (6%) |    13867 (6%) |   29168  (6%)
Low   |   234167(94%)|   231101(94%) |  465268 (94%)
Total |   249468(50%)|   244968(50%) |  494436(100%)

1. Subpopulation of size 23,532
Context = State: CA, Race: White
p-value = 2.31e-24 ; NMI = [0.0051, 0.0203]
Price | Income <$50K | Income >=$50K |       Total
High  |     606 (8%) |      691 (4%) |   1297  (6%)
Low   |    7116(92%) |    15119(96%) |  22235 (94%)
Total |    7722(33%) |    15810(67%) |  23532(100%)

2. Subpopulation of size 2,198
Context = State: NY, Race: Black, Gender: Male
p-value = 7.72e-05 ; NMI = [0.0040, 0.0975]
Price | Income <$50K | Income >=$50K |      Total
High  |      52 (4%) |        8 (1%) |   60  (3%)
Low   |    1201(96%) |      937(99%) |  2138 (97%)
Total |    1253(57%) |      945(43%) |  2198(100%)
... more entries (sorted by descending NMI) ...
```

Fig. 2: **Sample Association Bug Report.** Shows the full population and two highly-affected subpopulations for a Testing investigation of suspected disparate impact in a Staples pricing simulation.

located within 20 miles of a competing OfficeDepot store. Fig.2 shows part of FairTest's bug report generated by testing for suspected differential pricing based on income; similar reports were generated when testing on other attributes, such as race (see Appendix A.1). The report lists some of the statistically significant associations discovered between protected attribute '`income`' and output '`price`' in various subpopulations. The strength of the association is measured with *normalized mutual information* (NMI), one of several measures of statistical dependence implemented in FairTest (details are in §4.2).

The report shows three populations: the global population is first, followed by the two discovered subpopulations exhibiting the strongest disparities (highest NMI). The subpopulations are defined by user attributes: white people in California (first subpopulation) and black men in New York (second subpopulation). For each (sub)population, FairTest reports various statistical information: a p-value (a measure of statistical significance, with a value below 5% generally considered statistically significant), a confidence interval for the NMI metric, and a *contingency table* that summarizes the frequency distribution of the outputs over the (sub)population.

The contingency tables are particularly relevant for interpretation of the results. Here is how to read the entries in these tables. Focusing on the first subpopulation: among California's white population ($23{,}532$ people in our test set), $7{,}722$ (or 33%) have an income below \$50K. Out of these $7{,}722$ users, 606 (or 8%) were given the high price and the rest (92%) were given a discount.

The programmer can then interpret the report as follows: "At global U.S. population level, the disparate-impact of my pricing algorithm against lower-income people is nearly zero (NMI is close to zero for the global population, and the contingency table shows that low-income and high-income users receive higher prices in roughly similar proportion, 6%). However, the disparate impact is much stronger among white people in California (first subpopulation), where about 8% of lower-income people get higher prices vs. only 4% of higher-income people. Strong disparate impact also exists for black men in New York (second subpopulation), where 4% of lower-income black men get higher prices vs. 1% for higher-income black men." As remediation, the programmer might decide to alter her pricing scheme, e.g., to disable price tuning in affected regions, or to take into account publicly available statistics from various areas when deciding prices for people from those areas.

3.2 Architecture

Returning to the architecture (Fig.1(a)), FairTest expects three types of user attributes as inputs: (1) *Protected attributes*, $\boldsymbol{S}$, are discrimination-sensitive features (*e.g.,* race, gender, age) on which FairTest will look for associations. (2) *Context attributes*, $\boldsymbol{X}$, are dimensions along which FairTest will split the user population to identify smaller contexts with strong associations between outputs and protected attributes. These include user attributes that the programmer is knowingly using in his application (*e.g.,* location in the Staples pricing, or health history in the health application) and may also include protected attributes. (3) *Explanatory attributes*, $\boldsymbol{E}$, are user properties on which the developer deems it acceptable to differentiate, even if that leads to apparent discrimination on protected attributes. FairTest will explicitly avoid looking for bugs defined by these attributes. Explanatory attributes (described in §4.6) can be used to debug previously found association bugs (e.g., to rule out potential causes) or to distinguish unintended side-effects from consequences of an application requirement. While at least one protected attribute must be specified, explanatory and context attributes may be omitted.

FairTest's core (FairTest box in Fig.1(a)), which analyzes all the inputs has four architectural components:

1. The *Association Metrics* module defines an extensible set of metrics for associations. We incorporate five default metrics (defined in §4.2), chosen to support a variety of applications and investigations. As we show, switching between investigation types in FairTest simply requires selecting a different association metric.
2. The *Association Context Discovery* module implements a generic mechanism to identify subpopulations that exhibit particularly high associations (as measured by an appropriate metric) between an output and a protected attribute. To ensure that the derived association contexts are meaningful and easily interpretable, our mechanism builds a decision tree that recursively splits the user population into smaller, well-defined groups, with increasingly strong associations. We call this mechanism *guided decision tree construction*, and it constitutes a core technical contribution in our paper.
3. The *Statistical Validation and Ranking* module computes the statistical significance and association strength for each subpopulation (context) derived by the Context Discovery module. It incorporates an extensible set of statistical measurements, each appropriate for a particular association metric. It further filters associations by statistical significance, ranks them by effect size, and produces the bug report.
4. The *Dataset Management* module collects the data input by the developer, and manages data holdout (*i.e.,* test sets) in order to guarantee statistical validity of FairTest's results across one or more investigations.

3.3 FairTest Algorithm

Fig.1(b) shows FairTest's basic algorithm, which combines all of its inputs and modules in four steps: Step 1: Given a dataset, $D = \{(\boldsymbol{S}, \boldsymbol{X}, \boldsymbol{E}, \boldsymbol{O})\}$, FairTest first splits it into a *training set*, D_{train} and a *testing set*, D_{test}. Step 2: For each protected attribute $\boldsymbol{S}_i$ in $\boldsymbol{S}$, the Association Context Discovery module uses D_{train} to split the user population into meaningful subpopulations based on context attributes, $\boldsymbol{X}$; its goal is to maximize the association between $\boldsymbol{S}_i$ and $\boldsymbol{O}$ in the discovered contexts, using our guided decision-tree construction algorithm (§4.3). The association metric used depends on the investigation type, the presence of explanatory attributes $\boldsymbol{E}$, and $\boldsymbol{S}_i$ and $\boldsymbol{O}$'s data types; by default, FairTest picks a suitable metric from those it supports (§4.2). For each discovered association, the *Statistical Validation and Ranking* module (§4.4) assesses the bug's validity on D_{test} using p-values and confidence intervals (CIs) for effect sizes. A p-value here results from testing for the null hypothesis that an association bug is not present. A small p-value supports the conclusion that the bug in fact exists. Step 3: We correct p-values and CIs to account for the *multiple comparisons problem* that arises when making many statistical inferences. Step 4: To prioritize developers' efforts, we filter and rank association bugs (§4.4) to produce a report that includes all statistically significant associations, starting with the most affected subpopulations.

4 Detailed Design

This section details FairTest's design. We first give a more formal definition of the notion of unwarranted association (§4.1) and then describe FairTest's architectural modules and investigation features (§4.2-4.7).

4.1 Unwarranted Associations

As discussed in §2.2, the algorithmic fairness literature contains numerous and fragmented definitions for what constitutes "fairness" or "discrimination." Existing definitions have limited applicability, miss important considerations of utility and application requirements, and often rely upon difficult to establish thresholds for what constitutes "discrimination". A key foundational contribution in this paper is thus our development of a generic and broadly applicable concept, *unwarranted associations*, which encompasses broad classes of unintended unfair or biased side-effects of data-driven algorithms.

We define an unwarranted association as *any statistically significant association between a protected attribute (such as gender or race) and an algorithm outcome (such as a price, a hiring decision, or an error rate) that cannot be explained by any factor that is deemed as acceptable (e.g., a natural user inclination toward some class of products, an application requirement, etc.).*

This definition is objective, broadly applicable, and rooted in legal discrimination practice. For example, in U.S. law, a ratio of 4/5 for hiring rates of two groups is generally considered discriminatory, but lower effects may also qualify if statistically significant (and higher effects may be ignored if statistically insignificant) [14]. In addition, the notion of explanatory factors encompasses the legal notion of *business necessity*, where differential treatment of two groups is deemed acceptable if the differentiation can be shown to arise as a consequence of

Metric	Description	When to Use
Binary Ratio / Difference	Compares probabilities of a single output for two groups.	Binary $\boldsymbol{S}, \boldsymbol{O}$
Mutual Information (MI)	General dependence measure for two discrete variables.	Categorical $\boldsymbol{S}, \boldsymbol{O}$
Pearson Correlation (CORR)	Measures linear dependence between two scalar variables.	Scalar $\boldsymbol{S}, \boldsymbol{O}$; often for *Error Profiling*
Regression	For labeled outputs, measure associations for each label.	High dimension $\boldsymbol{O}$; always for *Discovery*

Table 1: **FairTest's Canonical Association Metrics**.

a fundamental business need. Appendix C gives further context for our definition from U.S. discrimination law.

The notion of statistical association is general and objective: it is any relationship between two measured quantities that renders them statistically dependent [51]. There are many metrics to measure association, each best suited to different contexts. After review, we chose a canonical set that supports a wide range of use cases.

4.2 Association Metrics

Table 1 shows the five canonical metrics supported by FairTest. They can be split into three categories, based on the types of a protected attribute $\boldsymbol{S}$ and output $\boldsymbol{O}$:

• *Frequency Distribution Metrics:* The association between categorical $\boldsymbol{S}$ and $\boldsymbol{O}$ (with few possible values) can be represented as a contingency table that displays the frequency distribution of these variables. In prior work, such tables were used to define *ratio and difference metrics* for binary variables [7,16,20,30,31,38,42,47,48, 57], but these are difficult to extend to non-binary classes of protected attributes [12]. In general cases, we summarize the association with *mutual information* (MI), the standard information-theoretic measure of dependence. We use a normalized version of MI (NMI) to compare effects across multiple associations on the same variables.

• *Correlation:* Measuring dependence of scalar variables (*e.g.,* with MI [44]) is hard, so it is common to consider specialized relationships for such variables. *Pearson's correlation* measures the strength of linear associations between $\boldsymbol{O}$ and $\boldsymbol{S}$, which may exist even for non-linearly related variables. These measures are often robust and broadly interpretable [46]. Note that a finding of zero correlation does *not* imply independence. However, as our aim is not to verify independence, and as we value interpretability, Pearson's correlation is a natural fit.

• *Regression:* High-dimensional output spaces occur in many use-cases, such as for applications that assign tags or labels to users, where it is not known *a priori* which specific tag/label to test for associations (see the Discovery investigation examples in §2.1). For these, we introduce a metric based on *regression*. At a high level, we model the relationship between the protected attribute $\boldsymbol{S}$ and a large number of dependent output labels $\boldsymbol{O}$ with a regression model (logistic or linear). This yields a *regression coefficient* for each label, with which we can estimate that label's association with $\boldsymbol{S}$.

Algorithm 1 Association-Guided Decision-Tree Construction
We build increasingly specific contexts with increasingly stronger associations, by recursively splitting the data upon the user attribute that maximizes the average association over derived contexts. Contexts are defined by predicates $\mathcal{P}$ over attributes $\boldsymbol{X}$ (*i.e.,* a path in the tree).

```
Require: MIN_SIZE                                   ▷ Minimum size of a context
Require: MAX_DEPTH                                  ▷ Maximum tree depth
Require: METRIC                                     ▷ Association metric
  function FINDCONTEXTS($D = \{\boldsymbol{S}, \boldsymbol{X}, \boldsymbol{E}, \boldsymbol{O}\}, \mathcal{P} = \emptyset$)
    Create an association context $\mathcal{P}$
    if $|D| <$ MIN_SIZE or $|\mathcal{P}| \geq$ MAX_DEPTH then
      return
    end if
    for $\boldsymbol{X}_i \in \boldsymbol{X}$ do
      $\mathbb{D} \leftarrow$ partition of $D$ based on the value of $\boldsymbol{X}_i$
      if $\exists D_i \in \mathbb{D}$ : METRIC$(D_i) >$ METRIC$(D)$ then
        Score $\leftarrow \sum_{D_i \in \mathbb{D}}$ METRIC$(D_i)/|\mathbb{D}|$        ▷ Avg. association
      end if
    end for
    if no partition yields a higher association score then
      return
    end if
    $\boldsymbol{X}_{\text{best}}, \mathbb{D}_{\text{best}} \leftarrow$ partition with highest score
    for $D_i \in \mathbb{D}_{\text{best}}$ do
      $V \leftarrow$ values taken by $\boldsymbol{X}_{\text{best}}$ in $D_i$
      FINDCONTEXTS($D_i, \mathcal{P} \cup \{\boldsymbol{X}_{\text{best}} \in V\}$)
    end for
  end function
```

To measure associations in the presence of an *explanatory attribute* $\boldsymbol{E}$, we extend each of the above metrics to measure the *conditional association* of $\boldsymbol{S}$ and $\boldsymbol{O}$, given $\boldsymbol{E}$. We thus quantify the average association between $\boldsymbol{S}$ and $\boldsymbol{O}$ that remains after controlling for $\boldsymbol{E}$ (see §4.6).

4.3 Association Context Discovery

A powerful feature in FairTest is its ability to efficiently "zoom into" a user population to discover subpopulations particularly affected by association bugs. This is important because strong associations may manifest only inside smaller groups, even if no effects are observed at full population level. In prior work, finding such bugs has required uninformed exhaustive enumeration of meaningful subpopulations, leading to a number of contexts either exponential in the feature space [47,48] or linear in the user space [38]. These approaches raise two concerns: (1) They require making a large number of statistical inferences, thus providing only weak guarantees on the false discovery rate. (2) They sacrifice the ability to discover small subpopulations (*e.g.,* a few hundred users), even if these exhibit the highest associations.

To effectively identify and investigate hidden association bugs, we develop a novel partitioning scheme, called *guided decision-tree construction*, which efficiently finds subpopulations that exhibit the strongest associations. In contrast to prior work, our method generates only a *constant* number of contexts, while aggressively searching for the smaller, most affected populations.

Alg.1 shows our algorithm. Inspired by decision-tree learning [45], it takes a new perspective in our context. While traditional tree-learning mechanisms greedily optimize some measure of target *homogeneity* (*e.g.,* Gini impurity), our algorithm actively maximizes some *association metric* between protected attributes and outputs.

The algorithm works by selecting a splitting rule, based on an attribute $\boldsymbol{X}_i \in \boldsymbol{X}$, so as to split the dataset into subsets with highest average association between $\boldsymbol{S}$ and $\boldsymbol{O}$. If $\boldsymbol{X}_i$ is continuous, we split the available data into two subsets, based on some threshold; if $\boldsymbol{X}_i$ is categorical, we split the data into one subset per value of $\boldsymbol{X}_i$. We only consider a split if it yields at least one sub-context with a higher association than the one measured over the current population. We then recursively apply this process on each subset derived from the highest scoring split. This approach: (1) permits use of *any* association metric; (2) produces simply-defined and interpretable subpopulations; and (3) aggressively searches for subpopulations with strong associations using just scalable/distributable computations [39].

We additionally employ well-known techniques for preventing this tree construction from overfitting the training data [45], such as bounding the tree's depth and pruning very small subpopulations (< 100 members).

4.4 Statistical Validation and Prioritization

Having discovered contexts that exhibit potential association bugs, we must validate and prioritize them before reporting them to developers. Validation is needed because we explicitly built the contexts over a user sample (D_{train}) so as to maximize associations. We validate bugs on an independent sample, the test data (D_{test}).

We use distinct notions of significance for bug validation and prioritization. For validation, we use statistical significance based on *hypothesis testing*: a bug is significant if its manifestation in the test set is unlikely under the "null hypothesis" (*i.e.,* the association between $\boldsymbol{S}$ and $\boldsymbol{O}$ is null). This is quantified by the *p-value* for a test. For prioritization, we use *effect size*, *i.e.,* the actual value of the association metric, estimated by means of a *confidence interval* (CI). FairTest incorporates statistical methods for computing p-values and CIs for all metrics in §4.2; for small samples, we use generic *permutation tests* [15] and bootstraps [13] instead of approximations. We apply Holm-Bonferroni corrections [26] to the p-values and CIs to ensure their simultaneous validity.

With these concepts, bug report generation works as follows: We filter out contexts with corrected p-values >0.05. We rank the remaining contexts by the lower bounds of their corrected effect-size CIs. We only include a context (*e.g.,* white males in NY) if it exhibits a stronger effect than the larger populations (*e.g.,* males in NY) that contain it. Eventually, the report (1) lists all statistically significant associations (even weak ones, if they affect large subpopulations) and (2) first displays the most strongly affected subpopulations.

```
class DataSource(D, budget)   # Holds out one test set per budget unit.

class Investigation(DataSource,S,X,E,O,M={})   # Base investigation class.
    # M stands for association metrics.
class Testing(DS,S,X,E,O,M)   # Investigation subclass for testing.
class Discovery(DS,S,X,E,O,M,top_k)   # Subclass for discovery.
    # Takes in number of outputs to consider in each context.
class ErrorProfiling(DS,S,X,E,O,M,groundTruth)   # Subclass for error profile.
    # Takes ground truth for a predictive output.

train(Investigations,maxDepth=5,minLeafSize=100)
    # Derives putative association contexts for one/more investigations.
test(Investigations,conf=0.95)   # Tests and corrects all associations.
report(Investigations,conf=0.95,outDir)   # Filters, ranks, saves reports.

class Metric   # Abstract class for association metrics.
Metric.computeStats(data, conf)   # Calculates p-value and CI at given level.
```

Fig. 3: **FairTest API.** Data holdout (1st), investigation types (2nd), methods to run investigations (3rd), API to implement for metrics (4th).

4.5 Investigations

The FairTest API exposed to developers is shown in Fig. 3. A core *Investigation* class is subclassed by three specific types, *Testing*, *Discovery*, and *ErrorProfiling*, which constitute FairTest's primitives for unwarranted association detection. To run these, a developer first gathers a set of user attributes and application outputs. This data is encapsulated in a *DataSource* that holds out one or more test sets for successive investigations (see §4.7). We next describe how each investigation type works; §6.3 shows how we use them in real applications.

Testing: This investigation type, our simplest and most intuitive, is used to test for the presence and strength of *suspected* associations. We used it in §3.1 to test for disparate impact in Staples' pricing scheme. A developer provides a dataset $D = \{(\boldsymbol{S}, \boldsymbol{X}, \boldsymbol{E}, \boldsymbol{O})\}$. Using its guided tree-construction mechanism, FairTest first finds contexts with potential associations. By default, FairTest selects a suitable association metric for the data types. The developer calls the *train* method to initiate context discovery, optionally tuning parameters for the size and complexity of the resulting contexts. She calls *test* and *report* to validate discovered bugs and produce reports.

Discovery: In some cases, such as the discriminatory labeling in Google Photos (see §2.1), it may be hard to anticipate which particular algorithm outputs (*e.g.,* photo labels) may exhibit unwarranted associations. A *Testing* investigation is thus impractical, as one would need to test for associations on each label separately. *Discovery* lets developers search for associations over a large number of outputs simultaneously. The insight is to use the regression metric from §4.2 to efficiently estimate the strength of the association between protected attributes and each output label. We then select the labels that exhibit the strongest associations (the number *top_k* of labels to select is tunable), and test each label individually using an appropriately chosen metric. These regressions are executed at every step of the guided decision-tree recursion, so at the end *Discovery* simultaneously yields both subpopulations that are labeled differently by the algorithm and the labels offered differentially to each

```
Report of associations of O=Admitted on Si=Gender,
conditioned on explanatory attribute E=Department:

Global Population of size 2,213
p-value = 7.98e-01 ; COND-DIFF = [-0.0382, 0.1055]
Admitted | Female    | Male      | Total
No       | 615(68%)  | 680(52%)  | 1295 (59%)
Yes      | 295(32%)  | 623(48%)  |  918 (41%)
Total    | 910(41%)  | 1303(59%) | 2213(100%)

* Department A: Population of size 490:
  p-value = 4.34e-03 ; DIFF = [0.0649, 0.3464]
  Admitted | Female   | Male     | Total
  No       |  9(15%)  | 161(37%) | 170 (35%)
  Yes      | 51(85%)  | 269(63%) | 320 (65%)
  Total    | 60(12%)  | 430(88%) | 490(100%)

* Department B: Population of size 279:
  p-value = 1.00e+00 ; DIFF = [-0.4172, 0.3704]
  Admitted | Female   | Male     | Total
  No       |  3(30%)  |  93(35%) |  96 (34%)
  Yes      |  7(70%)  | 176(65%) | 183 (66%)
  Total    | 10 (4%)  | 269(96%) | 279(100%)

* ...  Departments C-F, all with high p-values ...
```

Fig. 4: **Disparate Admission Rates in the Berkeley Dataset.** Shows a *Testing* investigation with explanatory attribute $\boldsymbol{E}$ = Department. COND-DIFF is the binary difference metric (DIFF), conditioned on $\boldsymbol{E}$.

subpopulation. In this sense, *Discovery* requires little *a priori* knowledge of what could constitute an association bug or what subpopulations it might affect.

Error Profiling: This is a type of *Testing*, where the sought association is the uneven distribution of algorithmic errors among users. *ErrorProfiling* unveils which populations are most affected by algorithmic mistakes, which may help improve both the accuracy and fairness of the algorithm, as we show for our healthcare predictor (§6.3.1). *ErrorProfiling* takes as inputs algorithm predictions and the corresponding ground truth, and computes a suitable error metric to be tested for associations.

4.6 Debugging

While FairTest's primary goal is detection of unwarranted associations, we considered it important to provide *some* basic support for debugging, or narrowing down the cause of these bugs. To this end, we introduce *explanatory attributes*, user properties that account for an unfair effect, or on which it is deemed acceptable to differentiate. For example, a company may decide that giving discounts to loyal customers is admissible even if this leads to a pricing bias against certain demographics.

A developer can use explanatory attributes in two ways: (1) She can define user properties $\boldsymbol{E}$ that are knowingly necessary for the application. FairTest then explicitly avoids deriving associations that are accounted for by these attributes, by measuring the dependence of protected attributes $\boldsymbol{S}$ and outputs $\boldsymbol{O}$ *conditioned* on $\boldsymbol{E}$. (2) After an initial investigation that reveals apparent unfair effects, she may *debug* these associations by specifying explanatory attributes that she believes are responsible (*i.e.,* confounders) for the observed behavior. FairTest then recomputes conditional association metrics over the *same* contexts discovered in the first investigation.

To illustrate the first use-case of explanatory attributes, we examine the Berkeley graduate admissions dataset, which contains admission decisions and gender for 4,425 applicants [2]. As mentioned in §2.2, this data exhibits a paradoxical effect: at full university level, admissions appear to disfavor women, yet this bias is not reflected in any department. We show how an analyst could use FairTest to measure gender-disparities in admission rates, while allowing that each department may have different gender demographics and admission rates.

The analyst defines '`department`' as an *explanatory attribute*, to instruct FairTest to look for associations only among applicants of the same department. She then runs a *Testing* investigation. The report (Fig.4) clearly illustrates the paradox: Over the full population, only 32% of female applicants are admitted versus 48% of male applicants. Yet, the only department with a significant gender disparity in admission rates (department 'A') actually favors women. Incidentally, the difference in admission rates *conditioned* on an applicant's department is found to not be statistically significant (the p-value is 0.798).

4.7 Dataset Management

Our decision to add debugging support in FairTest raises subtle issues with significant (and quite unexpected) implications for system design. To debug an association bug, developers must run multiple investigations, each informed by previous ones. The bug is detected in an initial investigation, after which the developer runs a series of other analyses (e.g., with explanatory attributes) to narrow down the cause of the effect. Because investigations use knowledge gained in previous steps, it is incorrect from a statistical perspective to validate them on the same test set as previous investigations.

The system design implications are significant. First, FairTest cannot be a stateless library; it must manage the dataset it is given for analysis across multiple investigations to enforce its correct use. Our current prototype achieves this by having developers specify a budget B (number of adaptive investigations they plan to run) upfront, when they supply a dataset; it then splits the dataset into B testing sets, each to be used for a single investigation. Only B investigations are allowed on the same dataset. The *DataSource* abstraction in FairTest's API (Fig.3) implements this functionality. More efficient approaches have been proposed [11], but they all lead to similar systems implications, restrictions, and interfaces.

Second, the number of investigations (B) a programmer can run on a given dataset is limited. This suggests that the best deployment for FairTest is one where testing sets are continuously replenished from production data. Care must be taken when investigating static datasets.

5 Prototype and Deployments

We implemented a FairTest prototype in Python, to be used as a standalone library or as a RESTful service. As a library, our prototype's workflows are designed to integrate with Pandas, SciPy's popular data analysis library, allowing developers to incorporate FairTest into their typical application testing process. Our service prototype enables continuous monitoring for association bugs in production systems. Developers register investigations with the FairTest service and route user attributes and outputs to it. FairTest runs investigations periodically and sends reports to the developers. The monitoring service continuously collects new data to replenish its test sets, thereby supporting larger numbers of investigations than the standalone library running on static datasets.

While FairTest is primarily designed for developers, we believe that it is also valuable for social-data analysts who wish to inspect datasets of public importance for signs of discrimination. For example, the American Civil Liberties Union (ACLU) and ProPublica have expressed interest in using FairTest to study social datasets such as imprisonment records, traffic law enforcement data, and school suspension data. They were particularly compelled by our subpopulation discovery mechanisms, which they believe can simplify and systematize their processes. We showcase both the developer and social-analyst use cases in §6.3 by running investigations on two data-driven applications and one public social dataset.

One key question is where programmers can obtain the data required to analyze their applications with FairTest. Generally, we believe developers should use whatever relevant user data they have available (which could include gender, race, location) and feed it into FairTest as protected or contextual attributes. Programmers can also use public datasets to emulate realistic application users (as we do for Staples, see §3.1). Finally, although not yet implemented, we envision a deployment model where a separate, trusted entity (such as the EFF, ACLU, or a census organization) collects a wider variety of user attributes, and offers the FairTest RESTful service to a range of privacy-conscious programmers interested in inspecting their applications for unwarranted associations.

6 Evaluation

Our evaluation addresses three questions: **(Q1)** Is FairTest effective at detecting association bugs? **(Q2)** Is it fast enough to be practical? and **(Q3)** Is it useful to identify and to some extent debug association bugs in a variety of applications? We use seven workloads:

- *One tightly controlled microbenchmark*, which we use to evaluate FairTest's bug detection abilities with a priory known ground truth for the associations.
- *Four data-driven applications* fed by public datasets: (1) a simulator of Staples' pricing scheme (as described by the WSJ report [52]) fed by U.S. census data; (2) a predictive healthcare application, based on a winning method and data from the Heritage Health Prize Competition [25]; (3) an image tagger based on Caffe [28], fed by ImageNet [9]; and (4) a movie recommender trained over the MovieLens dataset [6].
- *Two social datasets* – the Adult Census dataset [36] and the 1973 Berkeley Admissions dataset [2] – which have been used in prior algorithmic fairness work.

Table 2 shows workload information: number of users/attributes, investigations we ran, and metrics we used.

6.1 Detection Effectiveness (Q1)

Microbenchmark. Inspired by the Staples case, we create a microbenchmark that lets us control the strength and span of association bugs. We use U.S. Census [50] data for gender, race, and income to generate $\approx 1M$ synthetic users. We begin with a "fair" algorithm that randomly provides users with $\{0, 1\}$-output, independent of income. We then plant disparities in certain subpopulations (determined by location and race), so that income level (high or low) implies a difference in output proportions of size 2Δ. For example, we would give output "1" to 60% of high-income users and 40% of low-income users ($\Delta = 10\%$), for white users in California. For various subpopulation sizes and effect sizes, we inject 10 such randomly chosen discrimination contexts into our data and measure how many are discovered by FairTest.

Fig.5 shows FairTest's discovery rate as we increase population size and Δ. FairTest reliably detects strong disparities that affect at least a few hundred users, as well as effects as low as 2.5% in large contexts. However, low effects in small contexts often go undetected due to limited statistical evidence. In all cases, FairTest made zero false discoveries (finding a disparity that we did not introduce). Statistical testing lets us tightly control the false discovery rate: at a confidence level of 95%, we expect at most 5% false discoveries (after corrections).

Real-World Apps and Datasets. Table 2 reports the number of association contexts found by FairTest in each application. We show the number of potential bugs found by the guided decision-tree mechanism, the number of associations that are statistically significant after correcting for multiple testing, and the number of bugs reported to the developer (recall that we only report a context if it exhibits a higher unfair effect than the larger subpopulations that contain it). The size of the smallest reported context is also shown (the largest reported context is the full test-population). We do not have ground truth for these real-world workloads, but our experience inspecting these reports (detailed in §6.3) suggests that FairTest detects discrimination contexts of a variety of sizes, all of which appear accurate and revelatory for an investigator.

Results for the predictive healthcare application are for an experiment with a follow-up debugging investigation (see §6.3.1). As per §4.7, FairTest thus splits the dataset in three: a train set and two separate test sets. We found that FairTest would have reported the same bugs, had we used all the data for a single investigation (*i.e.,* with no

					Association Contexts			
Application	**Investig.**	**Users**	**Attributes**	**Metric(s)**	**Discovered**	**Validated**	**Reported**	**Size of Smallest Reported**
Microbenchmark	T	988871	4	NMI	n/a	n/a	n/a	n/a
Staples Pricing	T	988871	4	NMI	224	100	21	211
Predictive Healthcare	EP	86359	128	NMI,CORR	33	33	2	91
Image Tagger	D,T	2648	1	REG,DIFF	1	1	1	1324
Movie Recommender	D,T,EP	6040	3	REG,DIFF,CORR	54	19	11	223
Adult Census	T	48842	13	NMI	108	57	10	104
Berkeley Admission	T	4425	2	DIFF	1	0	1	2213

Table 2: **Workloads.** Investigations: *Discovery* (D), *Testing* (T), *ErrorProfiling* (EP). Metrics: normalized mutual information (NMI), correlation (CORR), binary difference (DIFF), regression (REG). For each application, we report the number of potential association contexts found by FairTest's context discovery mechanism, the number that were found to be statistically significant (p-value $< 5\%$), and the number of reported bugs.

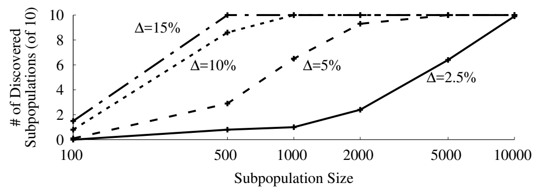


Fig. 5: **FairTest Effectiveness with Affected Subpopulation Size and Effect Strength (Δ).** Number of contexts discovered out of the ten we artificially inserted in 1M-user population. Average over 10 trials.

debugging). We further analyzed the effect of the budget B on the number of discovered and reported bugs for the Staples application. For budgets B of 2 and 3 (the train set and test sets each contain a $1/(B+1)$ fraction of the data), we discover 168 and 125 contexts, respectively; of these, we report 15 and 13 contexts, respectively. In both cases, the most affected subpopulation is the same as the one found for a budget $B = 1$. Thus, for this application, FairTest can allow at least one or two follow-up analyses, while preserving the main results reported to developers.

6.2 Performance (Q2)

We briefly discuss performance. Although its building blocks (decision trees, statistical tests) admit efficient and scalable implementations, our prototype does not incorporate all available optimizations. Still, we find that FairTest is fast enough for practical use. Fig.6 shows the analysis time for each of our applications (top numbers), broken down into: (1) the time spent on training to form association hypotheses, and (2) the time spent on testing and correcting these hypotheses. On a commodity laptop (4-core Intel CPU @1.7GHz, 8GB RAM), the total execution time ranges from 1-5 seconds for the smallest datasets to 80 seconds for the largest (Staples, with 1M users). For small datasets (Adult, Berkeley, Movies) we often use bootstraps and permutation tests to compute CIs and p-values in small contexts ($\leq$1000 users); these are expensive and subsume the training cost. For datasets that yield larger contexts, we use faster, approximate statistical methods, making the testing phase fast and the training phase proportionally more expensive.

6.3 Investigation Experience (Q3)

To assess FairTest's usefulness for developers, we investigated unwarranted associations in all real-world applications in Table 2. Our experience reveals that FairTest: (1) discovers insightful and interpretable associations and (2) assists programmers in debugging them.

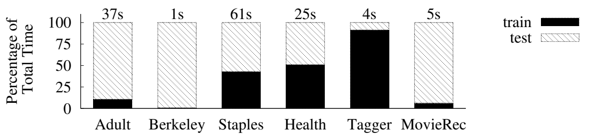


Fig. 6: **FairTest Performance.** Total FairTest analysis time (labels above bars) broken down into training and testing times (bars).

Specifically, we find interesting associations in all six applications and datasets, and we deem as really troublesome our findings in five of them: (1) *Staples pricing:* If applied uniformly in the U.S., the pricing scheme yields not only income-based disparity, but also racial discrimination, especially against Native Americans in Alaska. (2) *Predictive healthcare:* Although the predictor has good overall accuracy, its error unevenly affects older patients; in this case, FairTest enables us to narrow down the cause of the effect and identify remediations. (3) *Image tagger:* The tagger associates certain, potentially offensive labels with black people; all of these labels are clear errors. (4) *Movie recommender:* Older people get higher-rated movies than younger people; FairTest reveals a natural explanation – that older people tend to prefer war movies, which are rated higher than the action movies that younger people prefer – hence in this case, no bug occurs and no remediation is needed. (5) *Adult census:* FairTest confirms previously reported income disparities across gender and race and also discovers previously unreported effects, e.g., strong gender disparities among highly-educated people. (6) *Berkeley admissions:* FairTest reveals Simpson's paradox, as an explanation for an apparent gender discrimination.

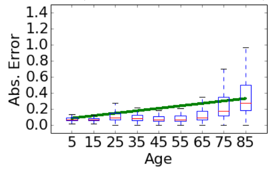
We detail three investigations: predictive healthcare, image tagger, Adult dataset. Others are in Appendix A.

6.3.1 Predictive Healthcare

Our predictive health application uses methods and data from the winners of the first milestone of the Heritage Health Prize Competition [25,43]. The random-forest based algorithm uses past healthcare claims to predict a user's number of hospital visits in the next year (predictions are for $\log(1 + \text{number of visits})$). The algorithm has low error overall (the average difference between the true and predicted number of visits is 0.42), but we want to study the error's distribution among users.

Our study gives an end-to-end view of how FairTest can be used to detect and debug unwarranted associa-

```
Report of associations of O=Abs. Error on S_i=Age:
Global Population of size 28,930
p-value = 3.30e-179 ; CORR = [0.2057, 0.2432]
```



```
1. Subpopulation of size 6,252
Context = Urgent Care Treatments >= 1
p-value = 1.85e-141 ; CORR = [0.2724, 0.3492]
```

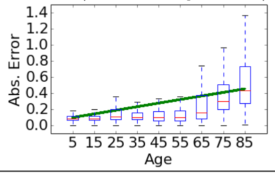
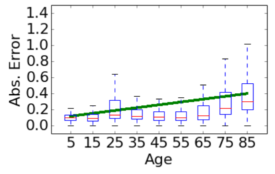


Fig. 7: **Error Profile for Health Predictions**. Shows the global population and the subpopulation with highest effect size (correlation). The plots visualize the correlation between age and prediction error, for predictions of $\log(1 + \text{number of visits})$. For each age-decade, we display standard box plots (box from the 1st to 3rd quantile with a line at the median and whiskers at 1.5 interquantile-ranges). The straight green line depicts the best linear fit over the data.

```
Report of associations of O=Abs. Error on S_i=Age,
conditioned on explanatory attribute E=Confidence:
Global Population of size 28,930
p-value = 1.26e-13 ; COND-CORR = [0.1050, 0.1597]
* Low Confidence: Population of size 14,481
  p-value = 2.27e-128 ; CORR = [0.1722, 0.2259]
```



```
* High Confidence: Population of size 14,449
  p-value = 2.44e-13 ; CORR = [0.0377, 0.0934]
```

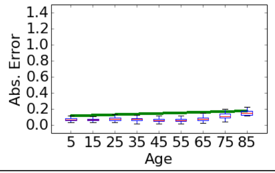


Fig. 8: **Error Profile for Health Predictions using prediction confidence as an explanatory attribute**. Shows correlations between prediction error and user age, broken down by prediction confidence.

tions, but also obtain hints for potential fixes. (1) We first discover an association bug: the application has much higher error rates for older than for younger people. (2) We investigate why the bug arises: the bias can be explained by lower prediction confidence for older people. (3) From there, we suggest potential fixes, such as only using high-confidence predictions. Our study consists of two investigations (detection and debugging), which we perform adaptively, each on its own test set.

Detection. We first use FairTest's *ErrorProfiling* to examine associations between the algorithm's prediction error and a user's age (scalar quantities, hence we use correlation). The report (Fig.7) shows the error/age correlations for the full user population and one subpopulation with higher effect. We visualize correlation with plots instead of contingency tables. Globally, prediction error grows with age (correlation is positive and the data shows a clear positive linear trend). This effect is strongest for patients with prior urgent-care treatments. In that context, the average error for patients of age 61-99 is 1.07, compared to 0.33 for younger patients.

This finding is alarming, as such disparities could cause quantifiable harms if, e.g., the algorithm is used to adjust insurance premiums (one of the competition's motivations [25]). Hence we wish to further investigate the *causes* of this accuracy loss for older patients, and get insights into how to fix this fairness (and accuracy) bug.

Debugging. We use FairTest's debugging abilities (explanatory attributes) to verify a plausible cause for the observed bias: The higher error for elderly patients could be due to the high *variance* of the prediction target (the number of hospital visits) for these users. To estimate the variance in a patient's target value, we train multiple predictors over random data subsets, and use these to infer prediction intervals for our algorithm's outputs [54]. The width of this interval is our estimate of the target's variance. Low variance means high prediction confidence.

We run a new *ErrorProfiling*, with prediction confidence as an explanatory attribute. Fig.8 shows the report. Conditioning on prediction confidence weakens the correlation in the full population. For users with low confidence, the correlation of error on age is still positive and significant, but for users with high confidence, the effect is almost entirely gone. We omit results for users with an urgent-care history, which are similar: the bias is almost gone for patients with high-confidence predictions.

Remediation Strategies. These results imply an immediate remediation strategy: when using this algorithm to, say, tune insurance premiums, one should consider the predictions' confidence. For example, one might decide to tune premiums only for high-confidence predictions. This would result in about half of the users in our dataset receiving customized premiums. One could also develop a scheme that weighs any price increase by prediction confidence. FairTest can then be used to test either of these approaches for disparate impact on the population.

6.3.2 Image Tagger

Our second scenario showcases FairTest's *Discovery* capability from the perspective of the developer of an image tagging system, who is willing to search for offensive labeling among racial groups. To illustrate the process, we inspect the labels produced by Caffe's [28] implementation of R-CNN [18], a ready-to-use image tagger, when applied to photos of people from ImageNet [9]. The tagger was itself trained on images from ImageNet with 200 tags, including images of people. We tag 1,405 images of black people and 1,243 images of white people with 5 labels each, and run a *Discovery* to find the 35 (top_k) labels most strongly associated with each race.

```
Report of associations of O=Labels on S_i=Race:
Global Population of size 1,324
* Labels associated with Race=Black:
Label   Black  White       DIFF          p-value
Cart      4%     0%   [0.0137,0.0652]  3.31e-05
Drum      4%     0%   [0.0095,0.0604]  3.83e-04
Helmet    8%     3%   [0.0096,0.0888]  2.34e-03
Cattle    2%     0%   [0.0037,0.0432]  4.73e-03

* Labels associated with Race=White:
Label         Black  White        DIFF            p-value
Face Powder     1%    10%   [-0.1339,-0.0525]  5.60e-12
Maillot         4%    15%   [-0.1590,-0.0575]  3.46e-10
Person         96%    99%   [-0.0563,-0.0042]  6.06e-03
Lipstick        1%     4%   [-0.0622,-0.0034]  1.03e-02
```

Fig. 9: **Racial Label Associations in the Image Tagger.** Shows partial report of a *Discovery* (top_k=35); the four most strongly associated labels (for the binary difference metric DIFF) are shown for each race.

Fig.9 shows part of FairTest's report. It lists the labels most disparately applied to images of black people (first table) and white people (second table); we show only 4 (of 35) labels per race. A developer could inspect all top_k labels and judge which ones deserve further scrutiny. In Fig.9, the 'cattle' label might draw attention due to its potentially negative connotation; upon inspection, we find that none of the tagged images depict farm animals. Moreover, black people receive the 'person' tag less often, thus the model seems less accurate at detecting them. Further work is needed to understand these errors; the model was possibly under-trained for these image types. While such analyses currently fall outside FairTest's scope, this example shows that FairTest is effective at providing "leads" for investigation. It will also help test the effectiveness of a remediation.

6.3.3 Adult Income Census Dataset

We next illustrate a second use case for FairTest: analysts studying discrimination in social datasets. We use the Adult dataset [36], which contains census data and income levels (under or over \$50K) for 48,842 U.S. citizens. Some discriminatory effects have been noted in prior algorithmic fairness works [16,20,32,38,56,57].

Fig.10 shows parts of FairTest's bug reports for *Testing* for income biases on race (top) and gender (bottom). We make three observations. First, FairTest confirms previously known race and gender biases in the full dataset: 88% of blacks have <\$50K-income compared to 75% of whites and 73% of Asians. Similarly, 89% of women have low income compared to 70% of men.

Second, FairTest reveals new insights into these biases. For race (top), black people are strongly disfavored among people younger than 42 working fewer than 55 hours a week – especially for federal government employees. For gender (bottom), the groups where women are most disadvantaged are: (1) older people with 9-11 years of education and (2) (perhaps surprisingly) people with a higher education ($\geq$12 years of education). We are unaware of any prior works in the algorithmic fairness area that have reported these particularly strong biases upon inspecting this dataset.

Third, as shown by the first context in Fig.10, FairTest

```
Report of associations of O=Income on S_i=Race:
Global Population of size 24,421
p-value = 1.39e-53 ; NMI = [0.0063, 0.0139]
Income|    Asian|     Black|...|     White|       Total
<=50K |556(73%) |2061(88%) |   |15647(75%)|18640 (76%)
>50K  |206(27%) | 287(12%) |   | 5238(25%)| 5781 (24%)
Total |762  (3%)|2348(10%) |...|20885(86%)|24421(100%)

1. Subpopulation of size 341
Context = Age <= 42, Hours <= 55, Job: Fed-gov
p-value = 3.24e-03 ; NMI = [0.0085, 0.1310]
Income|  Asian|   Black|...|   White|    Total
<=50K |10(71%)|62(91%) |   |153(63%)|239 (70%)
>50K  | 4(29%)| 6  (9%)|   | 91(37%)|102 (30%)
Total |14 (4%)|68(20%) |...|244(72%)|341(100%)

2. Subpopulation of size 14,477
Context = Age <= 42, Hours <= 55
p-value = 7.50e-31 ; NMI = [0.0070, 0.0187]
Income|    Asian|     Black|...|     White|       Total
<=50K |362(79%) |1408(93%) |   |10113(83%)|12157 (84%)
>50K  | 97(21%) | 101  (7%)|   | 2098(17%)| 2320 (16%)
Total |459  (3%)|1509(10%) |...|12211(84%)|14477(100%)
```

```
Report of associations of O=Income on S_i=Gender:
Global Population of size 24,421
p-value = 1.44e-178 ; NMI = [0.0381, 0.0540]
Income|    Female|       Male|       Total
<=50K |7218(89%) |11422(70%) |18640 (76%)
>50K  | 876(11%) | 4905(30%) | 5781 (24%)
Total |8094(33%) |16327(67%) |24421(100%)

1. Subpopulation of size 1,371
Context = 9 <= Education <= 11, Age >= 47
p-value = 2.23e-35 ; NMI = [0.0529, 0.1442]
Income|  Female|    Male|      Total
<=50K |423(88%)|497(56%)|  920 (67%)
>50K  | 57(12%)|394(44%)|  451 (33%)
Total |480(35%)|891(65%)|1371(100%)

2. Subpopulation of size 6,791
Context = Education >= 12
p-value = 3.71e-124 ; NMI = [0.0517, 0.0883]
Income|   Female|     Male|      Total
<=50K |1594(76%)|2156(46%)|3750 (55%)
>50K  | 492(24%)|2549(54%)|3041 (45%)
Total |2086(31%)|4705(69%)|6791(100%)
```

Fig. 10: **Disparate Impact Reports on Race (top) and Gender (bottom) in the Adult Income Dataset.** Shows the full population and two subpopulations with higher disparate effects.

is capable of revealing even small contexts that show particularly strong disparate effects. This capability has attracted ACLU's attention in particular, who wish to "zoom into" their datasets to identify populations under particular distress.

7 Conclusion

In a world where traditional notions of privacy are increasingly challenged by the myriad of companies that collect and analyze our data, we must rely on those companies' responsible use of our data to ensure our fair and moral treatment. We have presented *FairTest*, a tool that helps responsible, privacy-conscious developers to thoroughly check their data-driven applications for unfair, discriminatory, or offensive user treatment. Designed for ease-of-use by developers, FairTest enables scalable, statistically rigorous investigation of *unwarranted associations* between application outcomes and sensitive user attributes, such as race or gender. Our study of six applications and datasets shows the broad utility of FairTest's three key investigation types: *Discovery* of association bugs, *Testing* of suspected bugs, and *ErrorProfiling*.

References

[1] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan. Adscape: Harvesting and Analyzing Online Display Ads. *WWW '14: Proceedings of the 23nd international conference on World Wide Web*, Apr. 2014.

[2] P. J. Bickel, E. A. Hammel, and J. W. O'Connell. Sex bias in graduate admissions: Data from Berkeley. *Science*, 187(4175):398–404, 1975.

[3] T. Book and D. S. Wallach. An Empirical Study of Mobile Ad Targeting. *arXiv.org*, 2015.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] L. Breiman. Looking inside the black box. Wald Lecture II, 2002.

[6] D. Brent J., K. Joseph, H. Jon, G. Nathaniel, B. Al, and R. John. Jump-starting movielens: User benefits of starting a collaborative filtering system with "dead date". Technical report, University of Minnesota, March 1998.

[7] T. Calders and S. Verwer. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010.

[8] A. Datta, M. C. Tschantz, and A. Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. In *Proceedings of Privacy Enhancing Technologies Symposium*, 2015.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[10] K. Dixit, M. Jha, S. Raskhodnikova, and A. Thakurta. Testing the lipschitz property over product distributions with applications to data privacy. In *Proceedings of the 10th Theory of Cryptography Conference*, pages 418–436, 2013.

[11] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

[12] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 214–226, New York, NY, USA, 2012. ACM.

[13] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[14] Equal Employment Opportunity Commission. Information on impact (§ 1607.4), Uniform Guidelines on Employee Selection Procedure, 1978.

[15] M. D. Ernst. Permutation methods: A basis for exact inference. *Statistical Science*, 19(4):676–685, 2004.

[16] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 259–268, New York, NY, USA, 2015. ACM.

[17] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[19] J. Guynn. Google photos labeled black people 'gorillas'. USA Today, July 2015.

[20] S. Hajian and J. Domingo-Ferrer. A methodology for direct and indirect discrimination prevention in data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 25(7):1445–1459, 2013.

[21] A. Hannak, P. Sapiezynski, A. M. Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson. Measuring personalization of web search. In *WWW '13: Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, May 2013.

[22] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring Price Discrimination and Steering on E-commerce Web Sites. *IMC '14: Proceedings of the 14th ACM SIGCOMM conference on Internet measurement*, 2014.

[23] M. Hardt. How big data is unfair. Understanding sources of unfairness in data driven decision making. `https://medium.com/@mrtz/how-big-data-is-unfair-9aa544d739de`, September 2014.

[24] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

[25] Heritage Provider Network. Heritage Health Prize Competition. `http://www.heritagehealthprize.com/c/hhp`, 2012.

[26] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):pp. 65–70, 1979.

[27] M. Jha and S. Raskhodnikova. Testing and reconstruction of lipschitz functions with applications to data privacy. *SIAM Journal on Computing*, 42(2):700–731, 2013.

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[29] M. Kabra, A. Robie, and K. Branson. Understanding classifier errors by examining influential neighbors. In *CVPR*, 2015.

[30] F. Kamiran and T. Calders. Classifying without discriminating. In *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, pages 1–6. IEEE, 2009.

[31] F. Kamiran, T. Calders, and M. Pechenizkiy. Discrimination aware decision tree learning. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 869–874. IEEE, 2010.

[32] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012.

[33] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[34] M. Lecuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. XRay: Enhancing the Web's Transparency with Differential Correlation . In *23rd USENIX Security Symposium (USENIX Security 14)*, San Diego, CA, 2014. USENIX Association.

[35] M. Lecuyer, R. Spahn, Y. Spiliopoulos, A. Chaintreau, R. Geambasu, and D. Hsu. Sunlight: fine-grained targeting detection at scale with statistical confidence. In *Twenty-Second ACM Conference on Computer and Communications Security*, 2015.

[36] M. Lichman. UCI machine learning repository, 2013.

[37] B. Liu, A. Sheth, U. Weinsberg, J. Chandrashekar, and R. Govindan. AdReveal: improving transparency into online targeted advertising. In *HotNets-XII: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM Request Permissions, Nov. 2013.

[38] B. T. Luong, S. Ruggieri, and F. Turini. k-NN as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 502–510. ACM, 2011.

[39] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *arXiv preprint arXiv:1505.06807*, 2015.

[40] A. Ng. Advice for applying machine learning. `http://cs229.stanford.edu/materials/ML-advice.pdf`, 2011.

[41] A. Ng. Machine learning system design. `https://d396qusza40orc.cloudfront.net/ml/docs/slides/Lecture11.pdf`, 2013.

[42] D. Pedreschi, S. Ruggieri, and F. Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 560–568. ACM, 2008.

[43] R. A. Phil Brierley, David Vogel. Heritage Provider Network Health Prize. Round 1 Milestone Prize. How We Did It – Team 'Market Makers'. `https://www.kaggle.com/wiki/HeritageMilestonePapers/file/Market%20Makers-Milestone1DescriptionV21.pdf`.

[44] B. Poczos, L. Xiong, and J. Schneider. Nonparametric divergence estimation with applications to machine learning on distributions. In *Uncertainty in Artificial Intelligence*, 2011.

[45] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[46] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):pp. 59–66, 1988.

[47] S. Ruggieri, D. Pedreschi, and F. Turini. Data mining for discrimination discovery. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(2):9, 2010.

[48] S. Ruggieri, D. Pedreschi, and F. Turini. Integrating induction and deduction for finding evidence of discrimination. *Artificial Intelligence and Law*, 18(1):1–43, 2010.

[49] L. Sweeney. Discrimination in online ad delivery. *Queue*, 11(3):10, 2013.

[50] United States Census Bureau. Easy stats. `http://www.census.gov/easystats/`, September 2015.

[51] G. Upton and I. Cook. A dictionary of statistics, 2008.

[52] J. Valentino-DeVries, J. Singer-Vine, and A. Soltani. Websites vary prices, deals based on users' information. *The Wall Street Journal*, December 2012.

[53] T. Vissers, N. Nikiforakis, N. Bielova, and W. Joosen. Crying Wolf?On the Price Discrimination of Online Airline Tickets. *Proceedings of the 7th Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*, pages 1–12, June 2014.

[54] S. Wager, T. Hastie, and B. Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research*, 15(1):1625–1651, 2014.

[55] X. Xing, W. Meng, D. Doozan, N. Feamster, W. Lee, and A. C. Snoeren. Exposing Inconsistent Web Search Results with Bobble. *Passive and Active Measurements Conference*, 2014.

[56] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 325–333, 2013.

[57] I. Zliobaite, F. Kamiran, and T. Calders. Handling conditional discrimination. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 992–1001. IEEE, 2011.

A Supplementary Investigations

A.1 Racial Biases in Staples Simulation

§3.1 describes our investigation of the income-based discriminatory side-effects of a Staples-inspired location-based pricing scheme. We observed a significant impact of lower-income people getting higher prices, particularly among white people in California. This section describes our investigation into the racial and gender effects of that same scheme. On race, we found that Native Americans are shown high prices in about 19% of the cases, three times more than the population average. Fig.11 shows the top of FairTest's report of pricing-race impact. Globally, the strongest negative impact of the location-based pricing scheme is for American Indians and Alaska Natives, a minority that accounts for less than 1% of the users in our dataset. At a finer grain, the subpopulation with the strongest disparities are low-income women in New York, with white users, as well as the very few native Americans in this category, being most affected by the location-based pricing. It is interesting to note that among low-income New Yorkers, disparities appear stronger for women than for men, implying that the geographical distribution of genders (for individual ethnic groups) is not completely uniform.

```
Report of associations of O=Price on S_i=Race:
Global Population of size 494,436
p-value = 2.31e-178 ; NMI = [0.0241, 0.0286]

Price|     Asian|      Black|   Hispanic|American Indian & Alaska Native|...|      White|     Total
High |  430  (2%)|  977  (2%)| 4013  (5%)|                       654(19%)|   | 22544  (7%)| 29168  (6%)
Low  |23244(98%)|60629(98%)|82652(95%)|                      2879(81%)|   |286877(93%)|465268 (94%)
Total|23674  (5%)|61606(12%)|86665(18%)|                      3533  (1%)|...|309421(63%)|494436(100%)

1. Subpopulation of size 7,337
Context = Gender: Female, State: NY, Income: <50K
p-value = 4.73e-102 ; NMI = [0.0936, 0.1573]

Price|   Asian|    Black|Hispanic|American Indian & Alaska Native|...|    White|    Total
High |  12 (2%)|  17 (1%)|  31 (2%)|                          5(14%)|   | 493(15%)| 566  (8%)
Low  |512(98%)|1482(99%)|1795(98%)|                        31(86%)|   |2848(85%)|6771 (92%)
Total|524  (7%)|1499(20%)|1826(25%)|                        36  (0%)|...|3341(46%)|7337(100%)
```

Fig. 11: **Disparate impact of Staples' pricing scheme across ethnic groups.** In the global population, American Indians and Alaska natives are negatively effected. For low-income women in New-York, white users are also strongly disadvantaged.

A.2 Movie Recommender Investigation

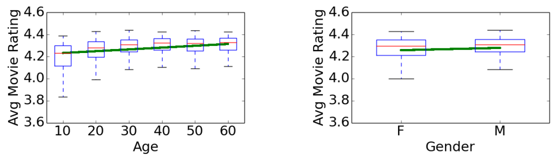
Omitted from the body of the paper is our investigation of the movie recommender, which revealed interesting but naturally explained effects, underscoring the importance of incorporating notions such as business necessity and utility when designing systems for detection and prevention of unfairness.

Our movie recommender is trained using the alternating least squares algorithm [33] and the MovieLens-1M dataset [6] (1M ratings provided by 6,040 users on a total of 3,900 movies). The ratings take values in $[1, 5]$, and each user has rated at least 20 movies. The dataset also includes user demographics (*e.g.,* age, gender) and movie metadata (*e.g.,* release date, genre). The test set is comprised of 10 randomly chosen ratings per user, and the rest of the data are used as the training set. The system is trained to model the kinds of movies users generally like. Furthermore, the system can be configured to recommend new movies and also predict the rating that a user will give a movie. For *ErrorProfiling*, we measure the root-mean-squared-error of our system's predicted ratings (for each user) over the test set.

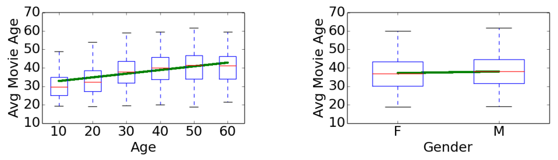
We show how programmers can combine FairTest's investigations to perform end-to-end explorations of associations. We refrain from making adaptive investigations, as the dataset is small. Instead, we set up all our investigations up front, and use FairTest to evaluate them over a single test set. Standard corrections for multiple testing guarantee the validity of the results.

Our first experiment is to *Test* for differences in simple characteristics of recommended movies (*e.g.,* a movie's average user rating – a proxy for popularity – and a movie's age in years since its release). We hypothesize that the *genre* of recommended movies could be linked to potential disparities in movie popularity. We thus also set up a *Discovery* to find which genres are associated with age and gender. Finally, we initiate an *ErrorProfiling*, to see how the system's prediction errors vary across user demographics, as this may account for certain associations. FairTest's API then allows us to *train*, *test*

```
Report of associations of O=Ratings on S_i=Age:
Global Population of size 3,020
p-value = 1.97e-15 ; CORR = [0.0907, 0.2055]

Report of associations of O=Ratings on S_i=Gender:
Global Population of size 3,020
p-value = 1.11e-03 ; CORR = [0.0106, 0.1274]
```

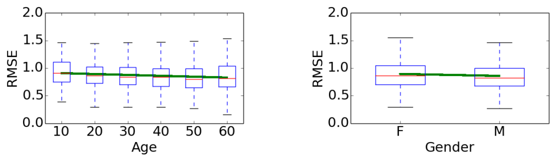


```
Report of associations of O=Movie Age on S_i=Age:
Global Population of size 3,020
p-value = 5.14e-62 ; CORR = [0.2435, 0.3457]

Report of associations of O=Movie Age on S_i=Gender:
Global Population of size 3,020
p-value = 2.46e-02 ; CORR = [-0.0079, 0.1038]
```



(a) Association Testing

```
Report of associations of O=RMSE on S_i=Age:
Global Population of size 3,020
p-value = 2.47e-04 ; CORR = [-0.1364, -0.0178]

Report of associations of O=RMSE on S_i=Gender:
Global Population of size 3,020
p-value = 6.82e-03 ; CORR = [-0.1204, -0.0016]
```



(b) Error Profiling

Fig. 12: **Associations for the movie recommender** (a) Correlation between user age and gender and the rating and age of offered movies. (b) Error profiling (RMSE for 10 ratings) across age and gender.

and *report* all three investigations simultaneously.

For the first investigation (see Fig.12(a)), FairTest reveals that movies recommended for women are overall a little less popular (but from the same time period) than those for men, and recommendations for older people are older and more popular than those for younger users. Our *Discovery* investigation further finds that women receive more recommendations for romantic movies, musicals and children movies, while men receive more action movies, thrillers and war films. Older users also receive many movies on war, while younger users get more action and crime films. These associations may offer a plausible explanation for the rating differences: war movies are among the most highly rated movies in the dataset, while the action and children genres typically score lower. Finally, when assessing the recommender's error distribution (Fig.12(b)), FairTest finds only small disparities, with men and older users getting slightly more accurate predictions overall than, respectively, women and younger users. This does not support our initial suspicion that disparities in the recommender's accuracy may account for differences in the popularity of recommended movies. Given additional data, a follow-up *Test* of association between popularity and gender or age using prediction error as an *explanatory attribute* might have uncovered more convincing evidence. Overall, these investigations illustrate that some associations we find in data-driven applications may have very natural and 'fair' explanations. It is thus crucial that systems for detecting and preventing unfair application behaviors be equipped to handle such confounding factors.

B Supplementary Related Work

We addressed related work compactly in §2.2. Because our work is inter-disciplinary, touching on fields including privacy (fairness and transparency) and machine learning, we provide further details about specific related works to give more context to people from different communities.

Algorithmic Fairness. Our work is most closely related to the growing field of *algorithmic fairness*, initiated by the work of Pedreschi et al. [42]. Their notion of fairness, based on manually imposed thresholds for binary association metrics, has been reused in a large number of works on discrimination detection [16, 38, 47, 48, 57] and prevention [7, 16, 20, 30, 31, 38, 57] (*i.e.,* the construction of 'fair-by-design' ML algorithms). Some authors have proposed explicit fairness definitions based on such binary metrics, *e.g.,* a-protection [20, 47, 48] or e-fairness [16]. More general association measures have also been considered, such as statistical parity (*e.g.,* [56], for binary protected attributes and arbitrary categorical outputs) and mutual information [32] (for general categorical attributes). As discussed in §2.2, the mechanisms presented in these works are always tied to the specific metric they consider, and thus often limited in scope and applicability.

More importantly, a majority of prior works has focused on assessing fairness strictly at the level of a full user population, thus sacrificing the ability to detect or prevent severe discrimination manifested in particular subpopulations [7, 16, 30–32, 57]. Those works that do integrate a notion of discrimination context [38, 47, 48] essentially resort to an exhaustive enumeration of all meaningful subpopulations, raising questions about the scalability, as well as about the false discovery rate of their approaches (corrections for multiple statistical testing are not considered for instance). Prior work also largely ignores plausible explanations for associations, thus treating any statistical dependence between protected attributes and outputs as a fairness violation [7, 16, 20, 30–32, 38].

Finally, a line of work initiated by Dwork et al. [12, 56] has considered discrimination prevention from an *individual's* point of view, essentially aiming at building classifiers that *treat similar people similarly*. A fundamental issue underlying this approach is that fairness is defined with respect to a socially agreed-upon *similarity metric* for users in a given context [12]. Constructing such a metric is highly non-trivial, especially when one only has access to information about individuals in a small sample, as opposed to *every individual from the population*. It is also non-trivial to test for the proposed notion of fairness on a sample (see [10, 27] for progress on this problem under very strong assumptions about the population).

Web Transparency. Our work also relates to the emerging field of *web transparency* [1, 3, 8, 21, 22, 34, 35, 37, 53, 55]. Although some works touch on discrimination and fairness (e.g., [8]), their setting is different: These works rely on *controlled, randomized experiments* that *probe* a service with different inputs (generally not real user profiles) and observe the effects on outputs, so as to identify and quantify Web services' use of personal data to target, personalize, and tune prices. Detection of unfair or unwarranted associations, as in FairTest, requires making inferences from application behavior on *real user profiles*, which may contain hidden correlations between inputs and sensitive values that would be unobservable with controlled experiments.

Debugging Machine Learning Applications. This header can mean at least two different things. The first are techniques to ensure a correct implementation of a well-specified learning algorithm (*e.g.,* a gradient descent algorithm for a particular objective function). The second are techniques to improve modeling and/or predictive performance of a classifier or predictor learned on training data. Here, we are primarily concerned with this second notion of debugging, although incorrect implementations of well-specified algorithms may also be diagnosed with similar techniques.

Machine learning practitioners typically use exploratory data analysis tools in somewhat *ad hoc* fashion to understand the data features and the modeling errors, at least as they manifest in training examples [24,40,41]. This includes conducting ablative analyses, examining feature dependency plots, and identifying commonalities among error cases. These techniques can also be applied to try to discover unwarranted associations between data features and protected attributes, but as we have argued in §2.2, they are not sufficient.

Some specific learning algorithms are more amenable to post-learning introspection. For example, in [29], the authors using bootstrap resampling to assess the influence of a single training example. A direct application of this idea is computationally prohibitive, but they show how to implement a fast approximation when using the "boosting" learning method [17]. As another example, in [5], the author shows that when using Random Forests learning method [4], the practitioner can "[look] inside the black box" (i.e., inspect the learned predictor) to perform many introspective analyses such as assessing the effects of features, clustering of training examples, and identification of outliers. Some of these analyses could be automated much like in FairTest, although the tree structures used in Random Forests are not constructed with the goal of finding unwarranted associations, and hence may miss many important effects.

C Relation to U.S. Discrimination Law

Our design of FairTest is informed by our own study of U.S. discrimination law (as legal non-experts) and informal discussions with discrimination law experts. Although the legal framework only applies to a few domains, such as housing, hiring, credit, housing, or health, FairTest extends legal concepts and applies them more broadly to many modern forms of data-driven decision making, such as tuning prices, labeling users' images, and personalizing healthcare processes. We discuss three aspects that have significantly influenced our design.

First, discrimination law encompasses both *differential treatment* (where intention, or causality, must be demonstrated before it can be prosecuted) and *disparate impact* (where demonstrating "significant harm" is sufficient, regardless of whether intention, or causality, can be established). FairTest's design is exclusively aimed at identifying disparate impact and leverages association, not causation, to measure that impact.

Second, key in discrimination law is the notion of *business necessity*: if a company can make the argument that the differential treatment or disparate impact in their processes appear as a result of an important business requirement, then they are exempt. For example, the hiring decisions of a trucking company may appear discriminatory against women, but if the bias can be explained by the fact that few or no women applicants have the requisite commercial driver's license, this apparent discrimination will be excused [48].

Third, establishing significant harm involves rather complex case-building processes. For example, in U.S. hiring law, a ratio of 4/5 for hiring rates of two groups is generally considered discriminatory, but lower effects may also qualify, depending on the case [14]. FairTest therefore does not attempt to impose a strict definition, threshold, or interpretation for what constitutes discrimination, but instead aims to reveal all statistically significant associations between an algorithm's outcomes and protected user groups.