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[1], Ari Juels[3], and Huang Lin[1]
[1]*École Polytechnique Fédérale de Lausanne* — [2]*Columbia University* — [3]*Cornell Tech*

October 9, 2015

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's 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, reliability, and performance 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 sensitive groups (*e.g.,* defined by race or gender). FairTest reports statistically significant associations to programmers as *association 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 tagging system.

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. It also enables powerful new applications, such as Google's image tagging system, which leverages tags entered by many users to automatically label every image on the Internet. Finally, data can improve business revenues by enabling effective product placement and targeted advertising.

Despite these undeniable benefits, integrating user data into applications can have unintended and detrimental consequences that are often difficult to anticipate for developers. A case in point is the Staples differential pricing case [46]. Staples' seemingly rational decision to adjust online prices based on user proximity to competitor brick-and-mortar stores led to pervasively 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 such risks will only increase as new kinds of personal and user-generated data – *e.g.,* collected through the Internet of Things – pass through increasingly complex machine learning algorithms, with associations and inferences that 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 [15]. Discriminatory online advertising has been found that associates ads for lower-paying jobs with women [6] and offensive, racially charged ads with black people [43].

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 reliability bugs. Such bugs may offend and even harm users, and cause programmers and businesses embarrassment, mistrust, and potentially loss of revenue. Unwarranted associations may also be symptomatic of a 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.

We present *FairTest*, a testing toolkit for data-driven applications that helps programmers test for and diagnose unwarranted associations. At its core, FairTest detects any associations between an algorithm's outputs (*e.g.,* prices or labels) and user subpopulations, including sensitive groups (*e.g.,* those defined by race, gender, or income level). It then reports any statistically significant associations as *association bugs*, filtered and ranked by their strength, statistical significance, and likelihood of being unintended side-effects rather than necessary effects. FairTest furthermore identifies both weak associations that affect large populations and strong associations that affect smaller populations. For example, our simulation of Staples's 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 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 without a priori knowledge of what bugs an application may present or what subpopulations these bugs may affect, (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 algorithm over a user population, that is, identifying any subpopulations with which erroneous algorithmic outputs are disparately associated. Using these capabilities, programmers can perform detailed investigations of association bugs, from discovery to diagnosis.

To make FairTest easier to use, we have integrated it into the evaluation framework of SciPy, a popular data analytics library. We used FairTest to test for disparate impact, discover offensive labeling, and profile algorithm errors in six applications and datasets, including: a simulation of Staples' pricing scheme fed by U.S. census data [45], a movie recommender, a predictive health application, and an image tagging system. We found association bugs in all cases, demonstrating the critical need for tools like FairTest to help uncover them in related families of data-driven applications.

We bring the following contributions to the space of algorithmic fairness, the closest domain to our work:

1. A *detailed study* of the algorithmic fairness literature that highlights the conceptual fragmentation, limited applicability, and scant experimentation to detect/prevent discriminatory associations (§2).

2. The *design and implementation* of FairTest, the first coherent, broadly applicable, extensible system that tests for unwarranted associations, such as disparate impact, offensive labeling, and error rate biases, in data-driven applications. FairTest simultaneously supports: (1) multiple fairness metrics as required by various applications and use cases, (2) efficient detection of unwarranted associations in user subpopulations, and (3) rigorous statistical result assessments (§3).

3. *Integration* of FairTest into SciPy in support of association bug discovery, testing, and error profiling (§4).

4. *Extensive experimentation* with FairTest's three investigation types on six real-world applications and datasets. We uncover association bugs in all cases, emphasizing the urgency of creating and deploying tools such as FairTest. Our experiments also showcase the usability of FairTest, exemplifying how programmers can compose FairTest's investigations to perform an end-to-end exploration of unwarranted associations, from discovery to diagnosis (§5).

5. FairTest's source code, released on paper publication.

2 Motivation

Our research aims to: (1) demonstrate the importance of testing for unwarranted associations in *any* data-driven application, 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 we agree that such sensitive applications should be closely inspected, we argue that *any* application that ingests and processes user data deserves scrutiny for association bugs. The following are examples of unwarranted associations that we or others have uncovered in various data-driven applications. We organize the examples based on the three investigation types presently supported by FairTest.

Example 1: *Discovery* of Association Bugs. This example shows the need for tools to aid developers in searching for questionable associations without a priori knowledge of what associations to consider. Association bugs can be hard to anticipate, as shown by the recent discovery of offensive labeling by Google's automatic image tagging system [15]. From the article [15]:

> *"Google has apologized after its new Photos application identified black people as 'gorillas.' On Sunday [a user] tweeted a screenshot of photos he had uploaded in which the app had labeled [him] and a friend, both African American, 'gorillas.' [...] [A Google representative] responded swiftly to [the user] on Twitter: 'This is 100% Not OK.' And he promised that Google's Photos team was working on a fix."*

In §5.2, we show how programmers can use FairTest's *Discovery* capability to proactively search for such offensive labeling (if it occurs consistently) *without* a priori knowledge of what might constitute offensive labels for certain populations. (Google engineers presumably did not know to scrutinize the assignment of "gorilla" tags in particular by their algorithms.) FairTest's discovery returns labels consistently given to certain subpopulations of users; the programmer could then judge labels as "appropriate" or "inappropriate" on a case-by-case basis.

Example 2: *Testing* of Association Bugs. This example shows the need for tools to aid developers in testing for specific suspected association bugs. We start from a 2012 Wall Street Journal (WSJ) article on unintended discriminatory behavior of Staples' online pricing algorithm [46]:

> *"[The investigators] found that the Staples Inc. website displayed different prices to people after estimating their locations. [...] If rival stores were within 20 miles or so, Staples.com usually showed a discounted price. [...] In what appears to be an unintended side effect of Staples' pricing methods [...] areas that tended to see the discounted prices had a higher average income than areas that tended to see higher prices."*

§5.2 shows how FairTest's *Testing* capability can be used to proactively test a specific set of potential upfront concerns or suspicions of disparate impact. We provide a step-by-step illustration of FairTest's use in this case using a simulation of Staples's purported pricing scheme fed with public census data.

Example 3: *Error Profiling* of ML algorithms. Our third example illustrates a different and broadly applicable type of investigation supported by FairTest: profiling the distribution of error in machine learning (ML) algorithms. Such investigations are a form of testing with the association bug being the disparate predictive error over a user population. A number of articles discuss potential sources of error bias in ML algorithms, such as behavioral differences between majority and minority groups, which can yield algorithms with good overall accuracy but high error rates for these minorities [19].

In §5.2, we further demonstrate this *Error Profiling* capability in FairTest. Using real-world data and a state-of-the-art, competition-winning approach from the Heritage Health Prize Competition, we trained a model that uses historical medical claims to predict whether a patient will be admitted to the hospital during the coming year. As foreshadowing, we found that while the model was fairly accurate overall (85% accuracy), its error disproportionately affects elderly patients, and is as high as 45% for older people with a history of "emergency" treatments.

2.2 System Requirements

The preceding scenarios illustrate the broad need for three types of investigations into unwarranted associations in data-driven applications. Other types of investigations may be needed for complete testing and debugging support for programmers. However, our experience suggests that *Discovery*, *Testing*, and *Error Profiling* are three core capabilities that enable detailed investigations of association bugs. To support these investigations effectively, systems and tools must meet the following requirements:

- *Generic and broadly applicable.* Tools must be applicable to many data-driven applications and investigation types. Their designs must therefore remain generic, and their implementations modular, to support extensions for unforeseen programmer needs.

- *Easy to use.* Tools must not assume that data-driven programmers are expert statisticians. They 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.

- *Address 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). Testing tools must discover both types of associations (and anything in between).

2.3 Prior Approaches

A sizeable literature on *algorithmic fairness* has focused on techniques for both *preventing* and *detecting* unfairness in machine learning and data mining algorithms. Existing work, however, provides neither tools that meet our preceding requirements nor adequate foundations to create them. Work on unfairness detection has focused on testing for specific discrimination bugs akin to those in Example 2; only one recent work [6] addresses discovery of such bugs (Example 1) in a very specific, ad-oriented setting; error profiling (Example 3) has never been considered. More importantly, our review has revealed important limitations in definitional foundations and in the development and evaluation of usable tools for data-driven programmers.

We closely studied 14 representative works from the algorithmic fairness field: seven on discrimination prevention [5, 9, 16, 22–24, 49], four on discrimination detection [6, 33, 38, 39], and three on both [13, 30, 50]. We make four observations about the field's state.

1. A proliferation of fairness criteria. The range of fairness criteria proposed in the literature is large and fragmented. It includes: metrics based on ratio proportions and differences [5, 13, 16, 22, 23, 30, 33, 38, 39, 50], a-protection [16, 38, 39], e-fairness [13], statistical parity [9, 49] and mutual information [24]. All these criteria measure some form of association of program *outputs* (*e.g.,* prices, errors) on *protected user attributes* (*e.g.,* race, gender). Our analysis of these criteria reveals two insights. First, different metrics are best suited for different situations and data types, and there is no best criterion for all use cases. Second, each proposed unfairness detection or prevention mechanism tends to focus on specific metrics that cover a limited range of cases. The majority of existing mechanisms apply only to binary protected features and outputs [5,13,16,22,23,30,33,38,39,50]. In this paper, we *integrate multiple, carefully chosen fairness criteria into a coherent system design with broad applicability.*

2. Limited consideration of discrimination contexts. A large part of the literature [5, 13, 22–24, 50] considers algorithmic fairness solely at full user population level. Yet prior work has shown that discriminatory effects in a population may differ from, and even contradict, those exhibited in smaller subsets, an effect known as Simpson's paradox [41]. A famous example of this paradox is the Berkeley admissions data [2]: University-wide admission rates appeared lower for women than men, yet when looking for similar effects in each department, either no bias or an inverse bias was revealed because women tended to apply to departments with lower acceptance rates. Unfortunately, the majority of prior works do not consider discrimination contexts as part of their system designs. Existing methods [30, 38, 39] exhaustively assess all possible contexts, leading to a number of subsets that is exponential in the feature space [38, 39] or linear in

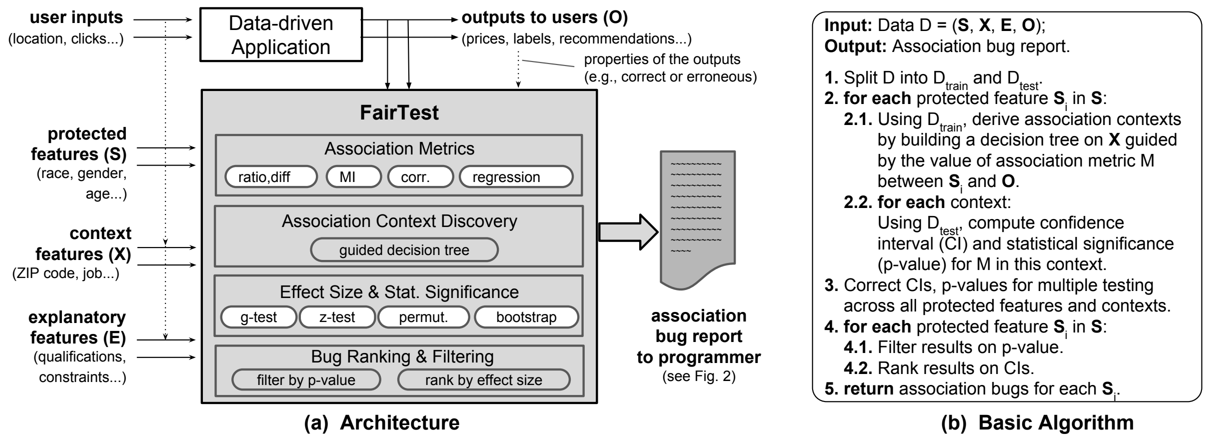


Fig. 1: **FairTest Architecture and Algorithm.** (a) Grey boxes denote FairTest components. Rounded boxes denote specific mechanisms. White rounded boxes denote extensibility points; transparent rounded boxes denote core, generic mechanisms in FairTest. (b) $\boldsymbol{S}$, $\boldsymbol{X}$, $\boldsymbol{E}$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

the user space [30]. Beyond their scalability issues, these works often fail to account for multiple-hypothesis testing for the many contexts they test, raising concerns about the statistical validity of their results. In our work, we develop a method for *efficiently and rigorously identifying of meaningful discrimination contexts.*

3. Use of thresholds rather than statistical significance. Most prior algorithmic fairness work defines fairness based on manually selected thresholds, *i.e.,* an algorithm is considered fair if a chosen bias metric falls below a threshold. Setting thresholds appears difficult (practically and morally) as it implies fixing a *permissible level* of discrimination [34]. In fact, in legal practice, it is customary to consider the *statistical significance* of a discriminatory effect in addition to a measure of that effect. For instance, in US hiring law, a ratio of 4/5 for the hiring proportions 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) [11]. Some authors have included confidence intervals for measures of differences and ratios in outcome proportions; these measures incorporate a notion of statistical significance, but the authors persisted in defining fairness strictly in terms of fixed thresholds. In addition, works that have reasoned about an algorithm's behavior in various subpopulations [30, 38, 39] have failed to correct effect sizes to account for multiple comparisons and thus provide only weak guarantees on the false positive rate of their discoveries. In contrast, our new method *rigorously and comprehensively detects statistically significant discrimination cases, and ranks these cases by their effect sizes.*

4. Limited experimentation and full-system experience. Most prior work describes limited experimental results, usually in the context of two datasets or less [5, 9, 16, 22–24, 38, 39, 50]. We believe that extensive experience with many applications and datasets is crucial for developing a robust and flexible system, one that can address real-world algorithmic fairness issues. Prior research also fails to cover system design issues, such as scalability and usability for developers. In our work, we conduct *an extensive experimental evaluation of FairTest with six real-world applications and datasets.*

2.4 Threat Model and Assumptions

FairTest is designed to aid developers in discovering association bugs. It is therefore intended to be used with *honestly designed* applications. Applications will not intentionally induce unwarranted associations over a target population and seek to conceal these associations from discovery by tools such as FairTest. While we aim to identify associations at smaller subpopulation level, we explicitly do not aim to discover tiny-scale associations (*e.g.,* at the level of an individual or a statistically insignificant group of individuals). For example, a movie recommender may behave poorly for a specific user, yet FairTest will not detect that.

Finally, FairTest does not currently support *adaptive data analysis* as would be required if embedded as unit tests in a build process. A programmer who blindly tweaks an application until it passes the unit tests may end up *overfitting* the data used in tests. A number of fixes are available to address this problem (*e.g.,* [8]), and we will incorporate these in future work.

3 FairTest Design

This section describes our design, starting with the system architecture and followed by a more detailed description of the four modules that comprise it.

3.1 Architecture

FairTest's generic and extensible architecture supports many different fairness criteria (*association metrics*) and investigation types with one coherent core. Fig.1(a) shows this architecture, indicating the core mechanisms and the extensibility points supported by FairTest.

Briefly, 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, $\boldsymbol{O}$, to the user. To use FairTest, the developer supplies it with a number of attributes (called *features*) about an applications' users, along with the outputs (or properties of the outputs) for those users. FairTest analyzes this data and returns an *association bug report* to the programmer.

In more detail, FairTest expects three types of user features as inputs to its analysis: (1) *Protected features*, $\boldsymbol{S}$, are the (sensitive) attributes on which FairTest will look for association bugs, such as race, gender, age. (2) *Context features*, $\boldsymbol{X}$, are dimensions along which FairTest will split the global population to identify smaller contexts in which the associations between outputs and protected features are strongest. These include user information 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 features. (3) *Explanatory features*, $\boldsymbol{E}$, are user properties on which the programmer deems it acceptable to differentiate, even if that leads to apparent discrimination on protected attributes. FairTest will explicitly avoid looking for bugs defined by explanatory features.

Basic Algorithm. Fig.1(b) shows FairTest's basic algorithm. The steps of the algorithm are as follows. 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 feature $\boldsymbol{S}_i$ in $\boldsymbol{S}$, select an appropriate *Association Metric* (§3.2) and apply the *Association Context Discovery* mechanism (§3.3) to D_{train} to identify meaningful subpopulations of users exhibiting putative association bugs. For each discovered association context, the *Statistical Significance and Effect Size* module (§3.4) 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 rejection of the null hypothesis and thus the conclusion that the bug in fact exists. Step 3: Correct p-values and CIs to account for the *multiple comparisons problem* that arises when making many statistical inferences. Step 4: Finally, to prioritize developers' efforts, filter and rank association bugs separately for each protected feature (§3.5) to produce a comprehensive report that includes all statistically significant association bugs, starting with the subpopulations that are most affected.

Investigations. We find the preceding architecture and algorithm to be flexible, supportive of many investigation needs and effective at producing insightful and directly interpretable results. To demonstrate, we considered a *Testing investigation* of suspected disparate impact of a Staples pricing simulation. We used U.S. census demographic statistics [45] to emulate users with realistic demographics and offered discounts to user located within 20 miles of an OfficeDepot store.

Fig.2 shows part of FairTest's bug report, beginning with a description of the dataset D. The association metric we use is *normalized mutual information* (NMI), a very general metric of association between two variables (defined in §3.2). It is applied here to protected feature `'income'` and output `'price'`. Two populations are shown: the global population is first, followed by the subpopulation with the strongest effect

```
S: {Income: (<$50K,>=$50K), Race: (White,Black,...)}
X: {State, City, Gender, Race}
E: {}
O: {Price: (High,Low)}

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%)|
+-----+------------+------------+------------+

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(32%)|   15810(67%)|23532(100%)|
+-----+------------+------------+-----------+
... XXX more entries (sorted by NMI desc.) ...
```

Fig. 2: **Sample Association Bug Report.** Shows the full population and highest-effect subpopulation for a Testing investigation of suspected disparate impact in a Staples pricing simulation.

(highest NMI) of all the derived subpopulations. For each context, FairTest reports the p-value, a CI for the NMI metric, and a *contingency table* providing the frequency distribution of the outputs over the population in that context.

With this information, the programmer can interpret the report as follows: "There is a statistically significant but extremely low disparate-impact effect of my pricing algorithm against lower-income people across the US. However, this effect is much stronger among California's white population, where about 8% of lower-income people get higher prices vs. only 4% of higher-income people." Incidentally, the next context (omitted in the figure) shows a similar, slightly weaker effect for black men in New York, where 4% of lower-income black men get higher prices vs. 1% for higher-income black men.

3.2 Association Metrics Module

No single metric is suitable for measuring the existence and strength of unwarranted associations in the full range of applications and use cases we wish to support. However, we can clarify the landscape of existing metrics by focusing on a canonical set of measures, shown in Table 1, that cover the vast majority of cases. We analyzed each metric along three dimensions: (1) applicability to classes of feature and output types, (2) interpretability, and (3) computational and statistical cost. §5.2 shows how we use various metrics in our investigations.

We next describe and justify our selection of canonical metrics. (We will drop the subscript i on $\boldsymbol{S}_i$ for clarity.)

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 ordinal variables.	Ordinal $\boldsymbol{S}, \boldsymbol{O}$; 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**.

• *Two Binary Measures.* A very natural metric for disparate impact compares the conditional probabilities of a binary output under two classes of a protected feature. For example, we may compare the proportion of users that receive a discount for users with either high or low income. Denoting the ranges of $\boldsymbol{O}$ and $\boldsymbol{S}$ by, respectively, $\{\boldsymbol{o}_1, \boldsymbol{o}_2\}$ and $\{\boldsymbol{s}_1, \boldsymbol{s}_2\}$, the *binary ratio metric* is $\Pr(\boldsymbol{o}_1|\boldsymbol{s}_1)/\Pr(\boldsymbol{o}_1|\boldsymbol{s}_2) - 1$, and the *binary difference metric* is $\Pr(\boldsymbol{o}_1|\boldsymbol{s}_1) - \Pr(\boldsymbol{o}_1|\boldsymbol{s}_2)$. These metrics are simple, interpretable, and used in numerous works on algorithmic fairness [5,13,16,22,23,30,33,38,39,50]. They can be efficiently estimated from a random sample by plugging-in maximum likelihood estimates of $\Pr(\boldsymbol{o}_1|\boldsymbol{s}_1)$ and $\Pr(\boldsymbol{o}_1|\boldsymbol{s}_2)$ (*i.e.,* plug-in estimator).

These metrics are easy to extend to non-binary outputs (*e.g.,* with total variation distance), but it is much harder to extend beyond binary classes of protected features (especially for estimation) [9]. In these cases, we instead assess dependence between $\boldsymbol{O}$ and $\boldsymbol{S}$ with fundamentally different metrics: mutual information and correlation.

• *Mutual Information* (MI) is an information-theoretic measure of statistical dependence. The mutual information between discrete-valued $\boldsymbol{O}$ and $\boldsymbol{S}$ is $I(\boldsymbol{O};\boldsymbol{S}) = \sum_{\boldsymbol{o},\boldsymbol{s}} \Pr(\boldsymbol{o},\boldsymbol{s}) \ln\left(\frac{\Pr(\boldsymbol{o},\boldsymbol{s})}{\Pr(\boldsymbol{o})\Pr(\boldsymbol{s})}\right)$. MI is non-negative, and zero if and only if $\boldsymbol{O}$ and $\boldsymbol{S}$ are independent (*i.e.,* when $\boldsymbol{S}$ has no effect on $\boldsymbol{O}$). MI measures the information content common to both $\boldsymbol{O}$ and $\boldsymbol{S}$. It is well-defined for arbitrary discrete outcome and protected feature spaces, and thus more widely applicable than binary metrics and their variants. The plug-in estimator for MI is also efficient as long as the ranges of $\boldsymbol{O}$ and $\boldsymbol{S}$ are not too large. A natural *normalization* of MI (NMI) divides the measure by the minimum of the Shannon entropies of $\boldsymbol{S}$ and $\boldsymbol{O}$.

MI ignores semantics of categorical values (*e.g.,* ordinal values like age and credit score), which may give unintuitive results. It can be defined for continuous $\boldsymbol{O}$ and $\boldsymbol{S}$ but is more expensive to estimate, especially in high dimensions [32, 35]. In these cases, we instead opt for a different metric: correlation.

• *Correlation.* A simple metric that is sensitive to ordinal and continuous values is *Pearson's correlation* between $\boldsymbol{O}$ and $\boldsymbol{S}$ (where the output $\boldsymbol{O}$ and protected feature $\boldsymbol{S}$ are scalar valued random variables.) Pearson's correlation measures the strength of a linear relationship between $\boldsymbol{O}$ and $\boldsymbol{S}$, which may exist even if $\boldsymbol{O}$ and $\boldsymbol{S}$ are non-linearly related; such linear relationships are typically robust and broadly interpretable [37]. Pearson's correlation is easily estimated using the well-known sample correlation. Unlike mutual information (and some non-linear dependence measures [14,44]), a finding of zero correlation does *not* imply that $\boldsymbol{O}$ and $\boldsymbol{S}$ are independent. However, because our aim is not to verify independence, and because we prefer interpretable findings of dependence, Pearson's correlation is a natural fit in many applications.

• *Regression.* None of the previous metrics is applicable or computationally tractable for high-dimensional outputs $\boldsymbol{O}$. Yet, large output spaces are common 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 association bugs. This is the case in the Discovery investigation examples described in §2.1. To support discovery of associations between a protected feature and an outcome from a large output space, FairTest relies on a novel application of *regression.*

We describe our approach for $\{0, 1\}$-valued $\boldsymbol{S}$ and applications that output a set of t labels, each in $\mathcal{L} = \{l_1, l_2 \ldots, l_d\}$ (so $\boldsymbol{O}$ takes values in $\mathcal{L}^t$). Let $b_1, \ldots, b_d$ be binary indicator variables for these labels, so that $b_i = 1 \iff l_i \in \boldsymbol{O}$. We model the conditional distribution of $\boldsymbol{S}$ given $\boldsymbol{O}$ by $\Pr[\boldsymbol{S} = 1 \mid b_1, \ldots, b_d] = \text{logistic}(\beta_0 + \sum_{i=1}^{d} \beta_i b_i)$, where the β_i are regression coefficients that can be viewed as measures of association between the labels and the protected feature. This approach extends to multi-valued or continuous $\boldsymbol{S}$ by replacing logistic regression with multinomial or linear regression.

Extensions. Despite our careful analysis of many metrics and selection of the preceding five as core metrics suitable for a wide range of association-bug explorations, we acknowledge that certain use cases may require specialized association measures. Hence, we designed FairTest to be extensible and expose a simple API for selecting specific metrics for each protected feature and investigation. By default, FairTest picks the "best" metric depending on the investigation, the protected feature, and the output types.

3.3 Association Context Discovery Module

A key problem with detecting association bugs is that they may manifest only in certain contexts, even if no effects appear over the full population. In prior work, detecting such contexts has required an expensive, broad-sweeping search for hidden associations [30, 38, 39]. To help developers identify, understand, and remove association bugs from their applications, FairTest incorporates mechanisms to efficiently discover hidden contexts that exhibit potential association bugs. Specifically, FairTest uses a general tree partitioning algorithm, which we call *guided decision tree construction*, an approach based on decision tree learning [36] that is novel to this setting.

Let $\boldsymbol{S}_i \in \boldsymbol{S}$ be the protected feature under investigation. We first select a splitting rule based on a feature in $\boldsymbol{X}$ that maximizes the association between $\boldsymbol{S}_i$ and $\boldsymbol{O}$ on the derived subsets (either the maximum or average over all subsets). We then recursively apply this process on each subset. This approach: (1) permits use of *any* association metric, (2) produces simply-defined and interpretable subpopulations, and (3) searches for subpopulations using scalable/distributable computations [31].

We further apply well-known techniques to prevent *overfitting* the training data [36], such as bounding the tree depth and pruning contexts that are simply too small (*e.g.,* <100 users). This helps avoid spurious associations that can be found in the training data but are nonexistent in the broader population. The contexts defined by this tree construction process are candidate bugs, which we validate on test data, a topic that we next discuss.

3.4 Computing Significance and Effect Size Module

After discovering contexts that potentially exhibit association bugs, the Computing Significance and Effect Size module validates them before reporting to the developer. After all, the associations were discovered only on a *sample* of users from a broader population. The module validates using an independent sample – the test data, D_{test} – taking cues from the statistical and legal communities to quantify the "significance" of an association bug.

We consider two notions of significance for association bugs. The first is based on *statistical hypothesis testing*: a bug is significant if its manifestation in the test set is unlikely under the "null hypothesis" (i.e., there is no association between the protected feature and application output as measured by an association metric). This is quantified by the *p-value* for a test. For all metrics in §3.2, well-known statistical methods exist for computing or approximating the p-value (*e.g.,* the *G-test* [42] for mutual information). For other metrics, or when the approximations are poor due to small sample sizes, we resort to generic *permutation tests* [12], which spend computational resources to gain accuracy with small test sets.

The second notion of significance is the *effect size*: the actual value of the association metric, or some derived quantity thereof, that relates the metric to application-specific harms. We reason about these quantities using *confidence intervals* (CIs) on effect sizes constructed using the test data. If the harm associated with particular outputs can be quantified (*e.g.,* monetary loss), metrics with simple semantics (*e.g.,* the binary difference) may directly translate to measures of harm. As before, for the metrics in §3.2, there are statistical methods for computing approximate CIs; in general cases, we can use bootstrap approaches [10] to produce these intervals. By default, the Computing Significance and Effect Size module chooses an appropriate method for computing CIs and p-values based on the metric and size of the test set.

For both notions of significance, we take steps to ensure their statistical validity when validating multiple association bugs. We apply standard Holm-Bonferroni corrections [21] to account for the *multiple comparisons problem* so that all our statistical inferences are simultaneously valid. These significance notions

```
class Investigation(D,S,X,E,O,M={})     # Base class for investigations.
    # M stands for association metrics.
class Testing(D,S,X,E,O,M)     # Investigation subclass for testing.
class Discovery(D,S,X,E,O,M,top_k)     # Subclass for discovery.
    # Takes in number of outputs to consider in each context.
class ErrorProfiling(D,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.computeSignificance(data)     # Calculates p−value.
Metric.computeEffectSize(data, conf)     # Calculates at given confidence level.
```

Fig. 3: **FairTest API.** Investigation types (top), methods to run investigations (middle), API to implement for new metrics (bottom).

provide the developer with rigorous and interpretable information about each association bug; they also enable automatic filtering and ranking of bugs, discussed next.

3.5 Ranking and Filtering Association Bugs Module

Programmers can easily become overwhelmed by irrelevant, redundant, or obviously explainable associations. The Ranking and Filtering Association Bugs module thus incorporates techniques for prioritizing discovered bugs. Our goal is to produce a bug report that: (1) contains all statistically significant associations (even weak ones, if they are validated on large enough subpopulations) and (2) first displays the subpopulations that appear most affected by a bug. To this end, the module filters out associations with (corrected) p-values >0.05 and ranks the remaining bugs by lower bounds on their effect sizes (from corrected CIs). We note that ranking bugs by their p-values does not provide the same properties: a small effect on a large subpopulation may have a smaller p-value than one with a considerably larger effect on a smaller subpopulation (see Fig.5).

Explanatory Features. In addition to these post-hoc rules for filtering and ranking, we proactively filter for *obviously explained or necessary associations*. A programmer may specify a set of *explanatory features*, or properties of a user on which it is deemed acceptable to differentiate. For example, a company may decide that giving discounts to loyal customers is always admissible even if it leads to a pricing bias against groups of less avid customers. Explanatory features can also encode the notion of *business necessity* supported by discrimination law, which justifies disparate effects given certain imperative business requirements [39].

FairTest lets developers declare user features $\boldsymbol{E}$ as necessary/explanatory for the application and explicitly avoids deriving associations that are accounted for by these features. To this end, it uses modified association metrics that measure the dependence of protected features $\boldsymbol{S}$ and outputs $\boldsymbol{O}$ *conditioned* on $\boldsymbol{E}$ (*e.g., conditional mutual information* $I(\boldsymbol{S};\boldsymbol{O}|\boldsymbol{E})$).

In addition to supporting business necessity and obvious explanations, explanatory features have another, compelling feature for FairTest: they enable *debugging* by letting developers eliminate plausible causes for association bugs. We have leveraged this feature in our investigations and will demonstrate its value in §5.2.

4 Prototype

We implemented a prototype of FairTest in Python, to be used either as a standalone library or as a RESTful service. It supports all three types of investigations, *Testing*, *Discovery*, and *Error Profiling*. As a library, our prototype's API and workflows are designed to integrate with the popular SciPy data analytics ecosystem, allowing programmers to seamlessly integrate FairTest-type investigations into their typical application testing process. As a service, our prototype enables continuous monitoring for association bugs in live,

Application	Investigations	Users	Features	Metrics
Microbenchmark	T	988871	4	NMI
Adult Census	T	48842	13	NMI
Berkeley Admission	T	4425	2	DIFF
Staples Pricing	T	988871	4	NMI
Health Prediction	EP	86359	128	NMI,CORR
Image Tagger	D,T	2648	1	REG,DIFF
Movie Recommender	D,T,EP	6040	3	REG,DIFF,CORR

Table 2: **Workloads.** Investigations: *Discovery* (D), *Testing* (T), *ErrorProfiling* (EP). Metrics: normalized mutual information (NMI), correlation (CORR), binary difference (DIFF), regression (REG).

production applications. We focus our evaluation of FairTest exclusively in the library implementation, the more mature of the two prototypes that we will release upon publication.

FairTest's API, shown in Fig. 3, provides four abstractions: a core *Investigation* class that defines an abstract investigation, and three specific investigation subclasses, *Testing*, *Discovery*, and *ErrorProfiling*. To use FairTest, a developer assembles a dataset of user attributes and application outputs, prepared in matrix form. The developer then instantiates the appropriate *Investigation* object, specifying which columns of the data correspond to algorithm outputs and contextual, protected, or explanatory features. In addition, he or she may specify association measures, one per protected feature in the investigation, which will override FairTest's defaults. Finally, the FairTest user calls the *train*, *test*, and *report* methods to run one or more investigations. These functions take as input various experiment parameters, such as the proportion of data to omit for testing (default=50%), the confidence level to use for p-values and CIs (default=95%), the maximum depth of the decision tree (default=5), and the minimum size of a subpopulation (default=100).

Discovery takes an additional parameter that limits the number of outputs considered in each context. *ErrorProfiling* takes as input the ground truth (*i.e.,* the true value for each prediction) and computes a suitable error measure, to be tested for associations on protected features.

Our prototype implements the five metrics described in §3.2. For binary measures and correlation, we use approximate methods to compute CIs and p-values for large sample sizes (>1000). Otherwise, we use bootstrapping and permutation tests. For normalized mutual information (NMI), we always use bootstrapping to compute CIs as the statistic's asymptotic behavior is less well studied. Finally, for *Discovery*, we use regression during training to efficiently identify k candidate associated labels, but we then switch to binary difference for testing, in order to compute each of these k labels' association on protected features, unconditionally from other labels.

5 Evaluation

In evaluating FairTest, we seek to answer three questions: **(Q1)** Is FairTest effective at detecting a wide range of association bugs? **(Q2)** Is it useful and usable for identifying and debugging association bugs in a variety of applications? and **(Q3)** Is it fast enough to be practical?

To answer these questions thoroughly, we assemble a diverse set of workloads, including microbenchmarks, real datasets, simulations, and real applications. Our goal is to exercise FairTest with various applications, metrics, and investigations to identify its strengths and weaknesses. We use seven workloads, which we plan to release as a benchmarking suite for future fairness efforts:

- *One tightly controlled microbenchmark*, which we use to evaluate FairTest's bug detection abilities with a priory known ground truth for associations.
- *Two well-known datasets* – the Adult Census dataset from the UCI ML-repository [28] and the 1973 Berkeley Admissions dataset [2], that have been used in prior algorithmic fairness work. We use these to confirm that FairTest can detect known associations (and in some cases discover previously unknown ones);
- *Four data-driven applications* that we feed with real data: (1) a simulator of Staples's pricing scheme (as described by the WSJ report [46]) fed by U.S. census data; (2) a predictive healthcare application, winner of the Heritage Health Prize Competition [20] and fed by the competition's data; (3) an image tagging

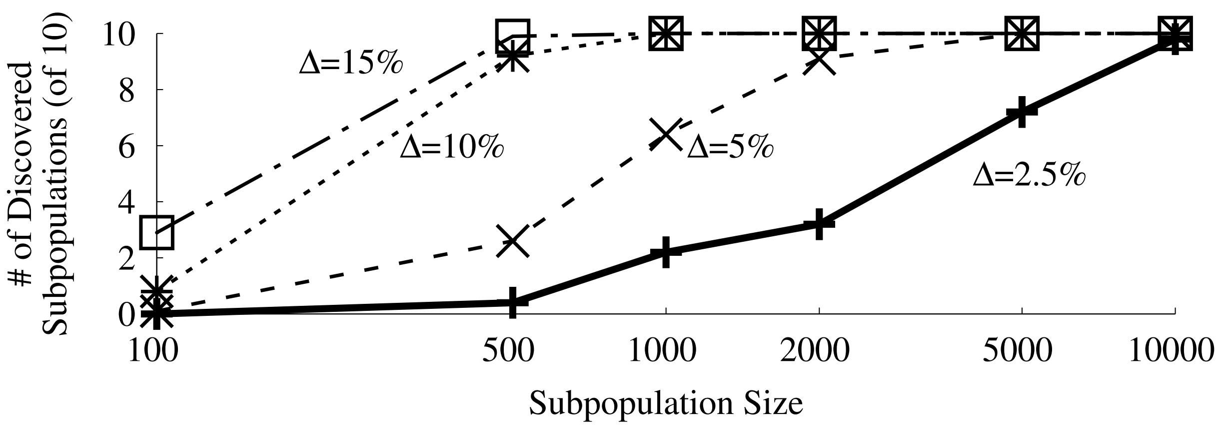


Fig. 4: **FairTest Effectiveness with Affected Subpopulation Size and Effect Strength.** Number of association contexts discovered by FairTest, out of the ten we artificially inserted in a population of 1M users. Subpopulation size grows on the x axis, effect strength grows with Δ. Shows averages over 10 trials.

system based on the OverFeat network [40], fed by images of people from the ImageNet database [7]; and (4) a movie recommender system based on collaborative filtering and fed by the popular MovieLens dataset [4].

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

5.1 Effectiveness of Bug Detection (Q1)

To evaluate FairTest's effectiveness at detecting bugs, we create a microbenchmark that lets us control the strength and span of association bugs. We use U.S. Census [45] demographics for gender, race, and income to generate $\approx 1M$ synthetic users. We begin with a "fair" algorithm that randomly provided users with $\{0,1\}$-output, independent of income. We then plant disparities in certain subpopulations of a given size (determined by location and race), so that income level (high or low) implies a difference in output proportions of size 2Δ. For various subpopulation and effect sizes, we inject 10 such randomly chosen discrimination contexts into our data and check how many are discovered by FairTest.

Fig.4 shows FairTest's discovery rate with increasing population size and for various values of Δ. Larger value of Δ means stronger disparity effect. FairTest reliably detects strong disparities that affect at least a few hundred users, as well as disparities as low as 2.5% over large contexts. However, low effects in small contexts often go undetected due to the limited statistical evidence available for FairTest to validate these cases. In all cases, FairTest made zero false discoveries.

5.2 Investigation Experience (Q2)

To assess FairTest's usefulness for data-driven programmers, we conduct six explorations of the associations produced by real-world applications and datasets. One application, the Staples pricing mechanism, is already discussed in §3.1, and additional results are in §A.3. Our investigations illustrate scenarios in which developers and analysts can use FairTest's bug finding and debugging capabilities to discover and analyze association bugs in applications. Our experience reveals that FairTest: (1) surfaces insightful and interpretable associations, and (2) assists programmers in debugging them.

Scenario 1: Confirming Income Inequality Effects in the Adult Dataset with *Testing*. We begin by analyzing the well-known Adult Income Census dataset [28], which consists of demographics and income level (under or over $50K) for 48,842 U.S. citizens. Although this dataset is not the product of a data-driven application, it has been analyzed in the algorithmic fairness literature, and certain disparate impact effects

```
Report of associations of O=Income on Si=Race:
Global Population of size 24,421
p-value = 1.34e-53 ; NMI = [0.0062, 0.0143]
+------+--------+---------...----------+-----------+
|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, Job: Fed-gov, Hours <= 55}
p-value = 6.24e-03 ; NMI = [0.0094, 0.1437]
+------+-------+-------...--------+---------+
|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%)|
+------+-------+-------...--------+---------+

4. Subpopulation of size 14,477
Context = {Age <= 42, Hours <= 55}
p-value = 7.21e-31 ; NMI = [0.0066, 0.0183]
+------+--------+---------...----------+-----------+
|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%)|
+------+--------+---------...----------+-----------+
```

(a) Disparate Impact on Race.

```
Report of associations of O=Income on Si=Gender:
Global Population of size 24,421
p-value = 1.39e-178 ; NMI = [0.0381, 0.0543]
+------+---------+----------+-----------+
|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 1,054
Context = {Education <= 9, Job: Sales}
p-value = 1.21e-22 ; NMI = [0.0530, 0.1739]
+------+--------+--------+----------+
|Income|  Female|    Male|     Total|
+------+--------+--------+----------+
|<=50K |464(95%)|416(73%)| 880 (83%)|
|>50K  | 22 (5%)|152(27%)| 174 (17%)|
|Total |486(46%)|568(54%)|1054(100%)|
+------+--------+--------+----------+

2. Subpopulation of size 6,791
Context = {Education >= 12}
p-value = 3.58e-124 ; NMI = [0.0504, 0.0875]
+------+---------+---------+----------+
|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%)|
+------+---------+---------+----------+
```

(b) Disparate Impact on Gender.

Fig. 5: **Disparate Impact Reports on (a) Race and (b) Gender in the Adult Income Dataset.** Shows the global population, the subpopulation with the strongest effect, and (a) its parent subpopulation or (b) the subpopulation with the second strongest effect.

have been identified [13, 16, 24, 30, 49, 50]. Our goal in this investigation is to show how an analyst can use FairTest to further study this and other social datasets.

We perform a *Testing* investigation to confirm biases of income based on race and gender (which are categorical features, hence our use of the NMI metric). Fig.5 shows parts of the bug reports for associations (a) income-race and (b) income-gender. We make three observations. First, FairTest confirms income race and gender biases at global population level: 88% of blacks have <$50K-income compared to 75% of whites and 73% of Asians. Similarly, 89% of women had <$50K-income compared to 73% of men. Second, FairTest finds interpretable contexts that give new insights into these societal biases. For example, Fig.5(a) reveals a much stronger race-income bias favoring white people among those under 42 working fewer than 55 hours a week – especially for federal government employees; Fig.5(b) shows that women with low education working in sales, and women with a higher education, are the most disadvantaged compared to men in similar situations. Finally, FairTest reveals small subpopulations that are nonetheless significant because they showed a strong association effect (*e.g.,* the first subpopulation in Fig.5(a)). Since it ensures simultaneous validity for all results in the report, the programmer can reliably compare these effect sizes, even for the small subpopulations.

Scenario 2: Investigating Disparate Impact in the Berkeley Dataset with *Explanatory Features*: Our second scenario provides a first look at FairTest's debugging capabilities. We examine the well-known Berkeley graduate admissions dataset, which contains admission decisions and gender for 4,425 applicants [2]. This data exhibits an unintuitive effect known as Simpson's paradox: at full university level, admissions appear to disfavor women, yet this bias is not reflected in any given department. We show how a hypothetical analyst can use FairTest to discover this effect in this and other datasets.

The process has two stages, shown in Fig.6. First, the analyst runs a *Testing* investigation. This investigation reveals a moderate but statistically significant bias over the full population (top box): only 32% of female applicants are admitted versus 47% of male applicants. Second, the analyst asks "Why?" and proceeds to investigate which department(s) might explain this bias. To do so, the analyst conducts another *Testing* investigation, this time using 'department' as an explanatory feature. FairTest's report for this investigation (bottom box) clearly illustrates the paradox in this dataset: only one department ('A') exhibits a significant difference in admission rates between men and women (in favor of women), while the other departments ('B', 'C', ..., 'F') exhibit no statistically significant bias. Therefore, the analyst may need to do further investigations to understand the cause of this paradox, with or without FairTest's help.

Scenario 3: Checking for Racial Labels in the Image Tagger with *Discovery.* Our third scenario show cases FairTest's *Discovery* capability from the perspective of a developer of an image tagging system, who is willing to search for offensive labels associated with sensitive groups. To illustrate the process, we inspect the labels produced by OverFeat [40], a ready-to-use image tagger, when applied to images of people from the ImageNet database [7]. We let OverFeat assign five labels to each of 1,405 images of black people and 1,243 images of white people, and then run a *Discovery* to identify the top_k=35 labels most strongly associated with a user's race. We note that OverFeat is only trained with images and tags of everyday objects and animals, but not people.

Fig.7 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); for clarity, we just show three labels per race. A developer may inspect all top_k labels to determine whether further scrutiny is warranted on a case-by-case basis. For example, the 'bulletproof vest' label is applied particularly to images of black people, even though none of them depict bulletproof vests. OverFeat also consistently associates female clothing tags with images of white people, and a large portion of these tags are erroneous. Further investigation is warranted to understand the causes of these errors; a likely conclusion one could draw is that OverFeat is under-trained for this population of images.

Scenario 4: Understanding Accuracy in the Predictive Healthcare App with *ErrorProfiling.* Our fourth scenario demonstrates FairTest's *ErrorProfiling* capability, which lets developers understand subtle biases in the error of a prediction application when ground truth predictions are available. We build a predictive healthcare system based on data and a winning method from the *Heritage Health Prize*

```
Report of associations of O=Admitted on Si=Gender:
Global Population of size 2213
p-value = 3.47e-13 ; DIFF = [-0.1945, -0.1122]
+--------+--------+---------+----------+
|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%)|
+--------+--------+---------+----------+<EOF>
```

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

Global Population of size 2213
p-value = 7.98e-01 ; COND-DIFF = [-0.0382, 0.1055]

* Department A: Global Population of size 490:
  p-value = 5.81e-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: Global 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. 6: **Investigating Simpson's Paradox in the Berkeley Admissions Dataset.** Results from an initial investigation, which reveals the apparent bias in the full population (top); results from a second investigation using explanatory feature $\boldsymbol{E}$ = Department (bottom). COND-DIFF is the binary difference metric (DIFF), conditioned on $\boldsymbol{E}$.

	Black	White	DIFF	p-value
Report of associations of O=Labels on Si=Race:				
Global Population of size 1324				
* Labels associated with Race=Black:				
	Black	White	DIFF	p-value
Mask	8%	2%	[0.0287,0.1026]	8.00e-07
Ski Mask	5%	0%	[0.0234,0.0807]	5.84e-08
Bullet Vest	5%	0%	[0.0137,0.0694]	1.56e-05
* Labels associated with Race=White:				
	Black	White	DIFF	p-value
Maillot	4%	21%	[-0.2309,-0.1171]	1.09e-21
Brassiere	2%	15%	[-0.1755,-0.0817]	4.29e-17
Miniskirt	2%	11%	[-0.1377,-0.0477]	1.49e-10

Fig. 7: **Race-based Label Associations in the Image Tagger.** Shows partial report of a *Discovery* (top_k=35) investigation; only top three most strongly associated labels are shown for each race.

Competition [20]. The application uses historical healthcare claims to predict a patient's number of visits to the hospital in the next year.

We use FairTest's *ErrorProfiling* to examine the distribution of the algorithm's absolute error based on patient age (scalar quantities; hence the use of correlation). FairTest's report (Fig.8) shows the error/age correlations for the overall population (a) and three subpopulations (b–d). The predictions for older people are patently more erroneous than for younger people, and this effect is strongest for patients who suffered a fracture and were admitted to the hospital for at least one emergency in past years (b). This association may translate into quantifiable harms if, for instance, the application is used to adjust insurance premiums. (Findings from a second *ErrorProfiling* investigation are given in §A.1.)

Scenario 5: Investigating Curious Effects in the Movie Recommender with *Testing*, *Discovery*, and *ErrorProfiling*. Our fifth scenario shows how programmers can compose FairTest's investigations to perform end-to-end explorations of associations. We build a movie recommender system using a dataset of 1M ratings from 6,040 users on 3,900 movies [4]. The dataset also includes user demographic features (*e.g.,* age, gender) and movie metadata (*e.g.,* release date, genre). We reserve 10 ratings per user as the test set; the rest comprise the training set. (More details are in §A.2.)

We are interested in how recommendations associate with user populations. We configure the system to output 50 new movies to each user, and *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). FairTest reveals that: (1) recommendations for women are more recent and less popular than those for men, and (2) recommendations for older people are older and more popular than those for younger people. (In this dataset, older movies tend to get higher ratings than younger movies.)

We hypothesize that the type of movies that are recommended account for disparities in recommended movie popularity. We thus run *Discovery* to determine what genres are associated with age and gender. FairTest finds: (1) women tend to receive more recommendations for romantic movies, musicals and children movies while men receive more action movies, thrillers and war films; (2) users of age >35 receive more recommendations for movies on war, while users of age ≤ 35 receive more action and crime films. These associations 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, we run *ErrorProfiling*, to see how the system's prediction errors are distributed in the population, as they also may partially account for associations found above. However, FairTest finds only small differences in accuracy, with men and older users getting somewhat more accurate predictions overall than, respectively, women and younger users. This result does not support our suspicion, although a follow-up *Test* of association between popularity and gender or age using prediction error as an *explanatory feature* could uncover some convincing evidence (à la Simpson's paradox).

Overall, this and the preceding scenarios show that FairTest reveals insightful associations and helps programmers investigate them.

```
Report of association of O=AbsoluteError on Si=Age:
(a) Global Population of size 43179
    p-value = 3.90e-179 ; CORR = [0.3219, 0.3494]
(b) Subpopulation of size 123
    Context = {Emergencies: [1, 4], Fractures: 1}
    p-value = 2.20e-03 ; CORR = [0.4006, 0.6835]
(c) Subpopulation of size 8177
    Context = {Emergencies: [1, 4]}
    p-value = 3.90e-179 ; CORR = [0.3792, 0.4385]
(d) Subpopulation of size 558
    Context = {Emergencies >= 5}
    p-value = 2.20e-03 ; CORR = [0.3740, 0.5296]
```

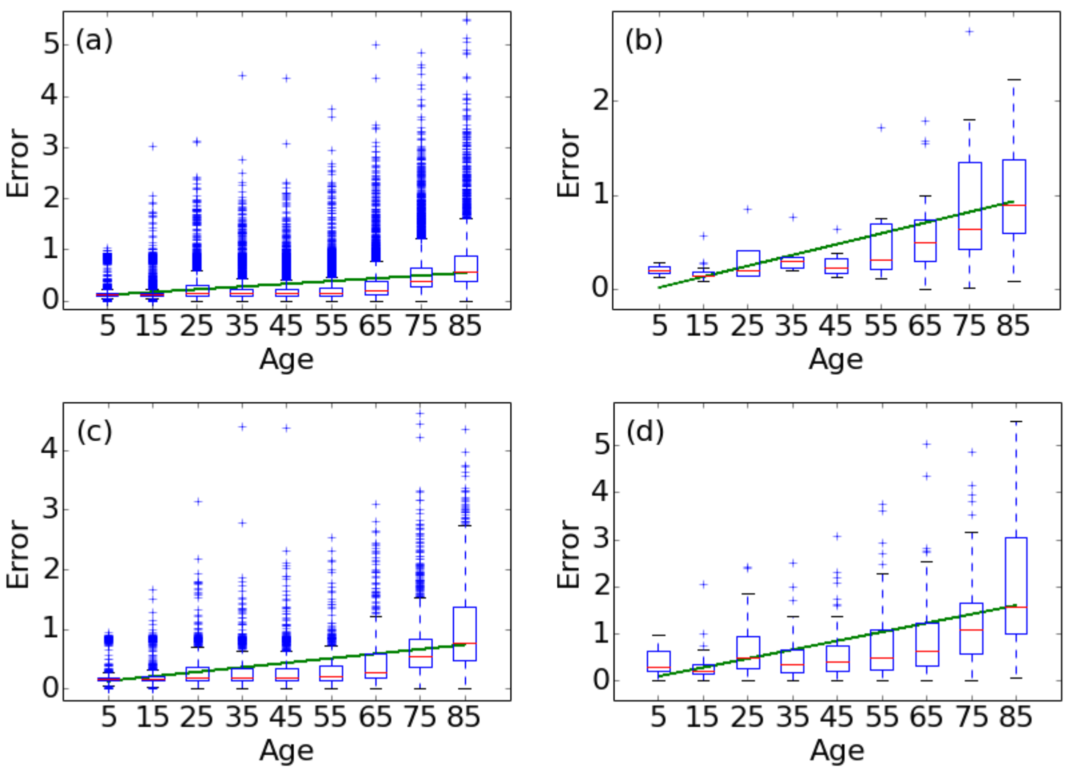


Fig. 8: **Error Profile for Health Predictions**. Shows the global population and 3 contexts with high effect size (correlation). Plots show the distribution of predictive error over age groups. The green line represents the best linear fit (least-squares) over the data.

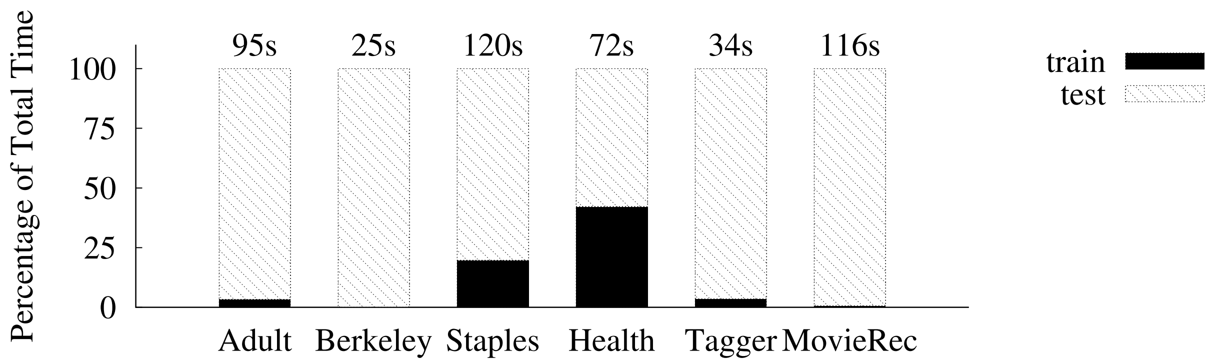


Fig. 9: **FairTest Performance.** Total FairTest analysis time (labels above bars) broken down into training and testing times (bars).

5.3 Performance (Q3)

We discuss FairTest's performance briefly. Although its building blocks (decision trees, standard statistical tests) admit of efficient and scalable implementations, our prototype does not currently incorporate the best available optimizations. We still find that our system is fast enough for practical use. Fig.9 shows FairTest's analysis time for each of our applications (top numbers), broken down in two components: (1) the time spent on training to form association hypotheses, and (2) the time spent on testing, correcting, filtering, and ranking these hypotheses. On a commodity laptop (4-core Intel CPU @1.7GHz, 8GB RAM), the total execution times range from 25 seconds for the smallest dataset (Berkeley) to two minutes for the largest (Staples, with 1M users). For the smaller datasets, Adult, Berkeley, Image Tagger, and Recommender (see Table 2 for dataset sizes), we often use bootstrapping and permutation tests for CI and p-value calculations in small contexts; these are expensive and subsume the cost of training. For Staples, a much larger dataset, we use faster, approximate computations of CIs and p-values, making the testing phase fast and the training phase proportionally more expensive. For the Health dataset, training is particularly expensive due to many user features (128), most serving as contextual features in the decision tree's generation. Further performance benchmarks are in §A.4.

6 Related Work

Apart from *algorithmic fairness* (reviewed in §2.3), the field of *web transparency* [1,3,6,17,18,26,27,29,47,48] is closest to our work; indeed, our team contributed two scalable and generic system designs [26, 27]. Web transparency is orthogonal to our work here, however, as it relies 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. Although some works touch on discrimination and fairness (e.g., [6]), the web transparency setting is also different from that in this paper. 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.

7 Conclusions

We have presented FairTest, a tool to help developers negotiate a world of increasingly complex application pipelines, machine learning algorithms, and data flows, as well as spreading impact of algorithms on users' lives. Designed with usability for developers in mind, FairTest enables scalable, statistically rigorous investigation of unwarranted associations in data-driven applications. Our study of a suite of six real-world applications and datasets demonstrates the broad utility of the three key investigation types currently supported in FairTest: *Discovery* of association bugs, *Testing* of suspected bugs, and *ErrorProfiling* for machine learning algorithms.

Although designed for developers, FairTest is largely a black-box analysis system. Our hope is that it will therefore not just serve application developers, but also help pave the way to more general algorithmic transparency, leading to tools that empower auditors, regulators, and users to act as watchdogs for pervasive fair data use.

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] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] 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.

[14] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In S. Jain, H. Simon, and E. Tomita, editors, *Algorithmic Learning Theory*, volume 3734 of *Lecture Notes in Computer Science*, pages 63–77. Springer Berlin Heidelberg, 2005.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] L. Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[33] 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.

[34] J. L. Peresie. Toward a coherent test for disparate impact discrimination. *Ind. LJ*, 84:773, 2009.

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

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

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

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

[39] 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.

[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*. CBLS, April 2014.

[41] E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 238–241, 1951.

[42] R. R. Sokal and F. J. Rohlf. *Biometry: The Principles and Practice of Statistics in Biological Research.* Freeman, second edition, 1981.

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

[44] G. J. Székely, M. L. Rizzo, and N. K. Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):pp. 2769–2794, 2007.

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

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

[47] 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.

[48] 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.

[49] 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.

[50] 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 Results

A.1 Predictive Healthcare Application (Scenario 3)

We give a few additional details and results from our investigation of the predictive healthcare application (Scenario 3 in §5.2). We use 25% of the Heritage Health Prize Competition's dataset to train our application, and reserve the remaining 75% for testing. The prediction is actually of log(1 + number of visits). Using the test data, we find that the application has root-mean-squared error in this prediction of 0.4, and also achieves 85% accuracy for the binary prediction of whether the patient visits the hospital.

Fig.10 displays FairTest's *ErrorProfiling* report for the binary healthcare classification task (will a patient visit the hospital next year?). Here we compare proportions of *false positives* (we predict a visit to the hospital for a healthy patient), *false negatives* (we fail to predict that a patient will visit the hospital) and *correct classifications* across age groups. Overall, our model's accuracy is about 90% for patients below 60 vs. only 76% for older patients. This disparity is even stronger for men who were treated for an emergency at least once and stayed at urgent care at least twice in the past. For these patients, the accuracy remains at 90% for the young but drops to 55% for the elderly. In particular, the false-positive rate is close to 0% for younger patients, yet over 35% for older patients in this context.

```
Report of association of O=FP/FN/True on Si=Age:
Global Population of size 43179
p-value = 4.50e-179 ; NMI = [0.0419, 0.0546]
+------+----------+----------+----------+-----------+
|Output|  Age 1-30| Age 31-60| Age 61-99|      Total|
+------+----------+----------+----------+-----------+
|True  | 9977(89%)|14616(90%)|11981(76%)|36574 (85%)|
|FN    |  953 (9%)| 1270 (8%)| 1720(11%)| 3943  (9%)|
|FP    |  266 (2%)|  326 (2%)| 2070(13%)| 2662  (6%)|
|Total |11196(26%)|16212(38%)|15771(37%)|43179(100%)|
+------+----------+----------+----------+-----------+

Subpopulation of size 736
Context = {Gender: Male, Emergencies >= 1,
           SDS Code = 0, Urgent Care >= 2}
p-value = 2.30e-04 ; NMI = [0.1153, 0.2616]
+------+---------+---------+---------+---------+
|Output|Age 01-30|Age 31-60|Age 61-99|    Total|
+------+---------+---------+---------+---------+
|True  | 209(93%)| 274(90%)| 114(55%)|597 (81%)|
|FN    |  15 (7%)|  26 (9%)|  21(10%)| 62  (8%)|
|FP    |   0 (0%)|   4 (1%)|  73(35%)| 77 (10%)|
|Total | 224(30%)| 304(41%)| 208(28%)|736(100%)|
+------+---------+---------+---------+---------+
```

Fig. 10: **Prediction error for the medical application.** Displays proportions of false negatives (FN), false positives (FP) and true predictions of a patient's hospital visit.

A.2 Movie Recommender (Scenario 5)

We next give a few details on the movie recommender investigation setup (Scenario 5 in §5.2) and illustrate the experience related there with results (Fig.11). Our movie recommender is trained using the alternating least squares algorithm [25] and the MovieLens-1M dataset [4] (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 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 over the test set.

As mentioned in §5.2, we first considered associations between a user's age or gender and a recommended movie's popularity or anciency. The top results in Fig.11(a) highlight weak disparities in the average ratings of recommended movies, with men and older users getting somewhat more popular movies than respectively women and younger users. Interestingly, a much stronger correlation appears between a user's age and the anciency of offered movies (bottom of Fig.11(a)). Men and women however tend to get movies of approximately the same time period. Finally, we profiled our recommender's error when predicting the ratings given by users (RMSE for 10 ratings per user). The results in Fig.11(b) again show weak, yet statistically significant, differences in prediction accuracy across age groups and genders, implying that our system is a little better at modeling the movie preferences of men and older users, than respectively women and younger users.

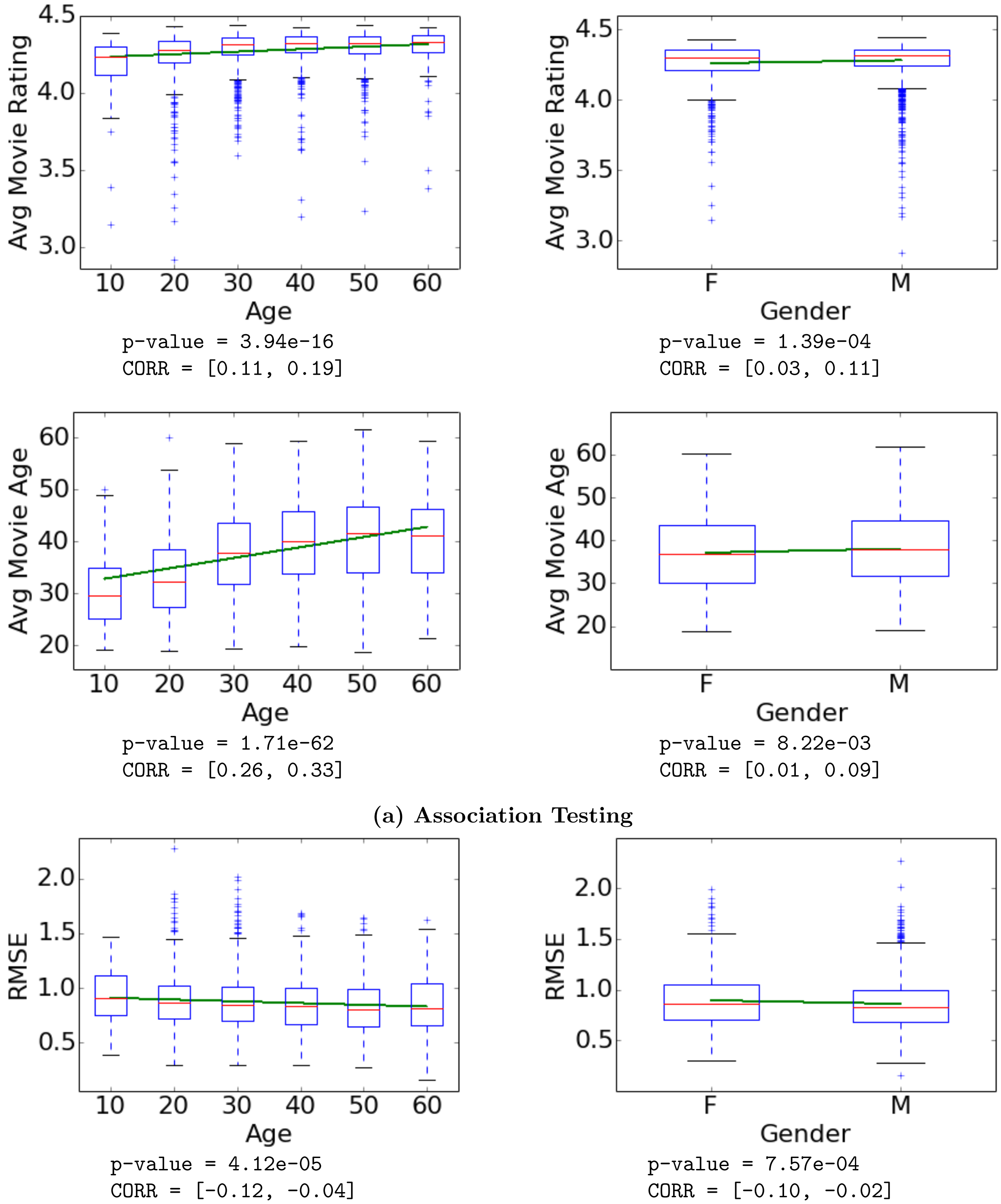
A.3 Staples Simulation

We next give further results from our *Testing* investigation of the Staples-inspired location-based pricing scheme (described in §3.1). We observed in Fig.2 a significant impact of lower-income people getting higher prices, particularly among white people in California. We additionally tested for disparate impact on gender and race. On race, we found that Native Americans are shown high prices in over 19% of the cases, three times more than the population average. Fig.12 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 female population 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.

A.4 Performance Microbenchmark

To microbenchmark FairTest's performance, we use the Staples dataset and artificially create additional pseudo-features. We duplicate the original demographic features and shuffle their values across the population. We produce bug reports for increasing number of features and input sizes (#of users) and evaluate the two main phases of FairTest's analysis time: *train* and *test*.

Fig.13 shows the results. Both times increase linearly with the number of features and the input size. The training phase (a) increases more rapidly, compared to the testing time (b) as the number of features grows. This is because testing time mostly depends on the number of subpopulations uncovered (and their sizes), which is not strongly dependent on the number of features. In contrast, building the association-guided decision tree obviously implies testing each feature for potential splits, which explains the increase.



(a) Association Testing

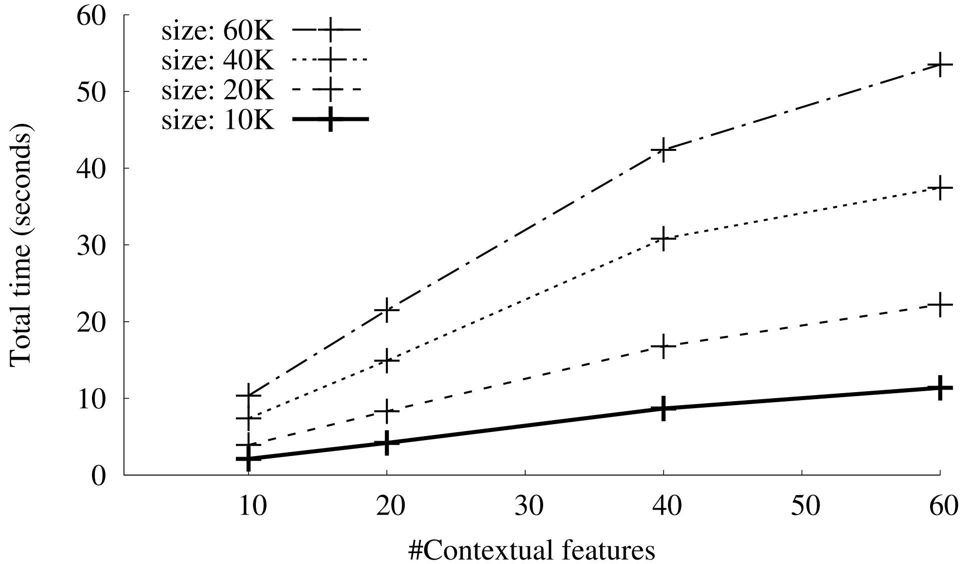
(b) Error Profiling

Fig. 11: **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.

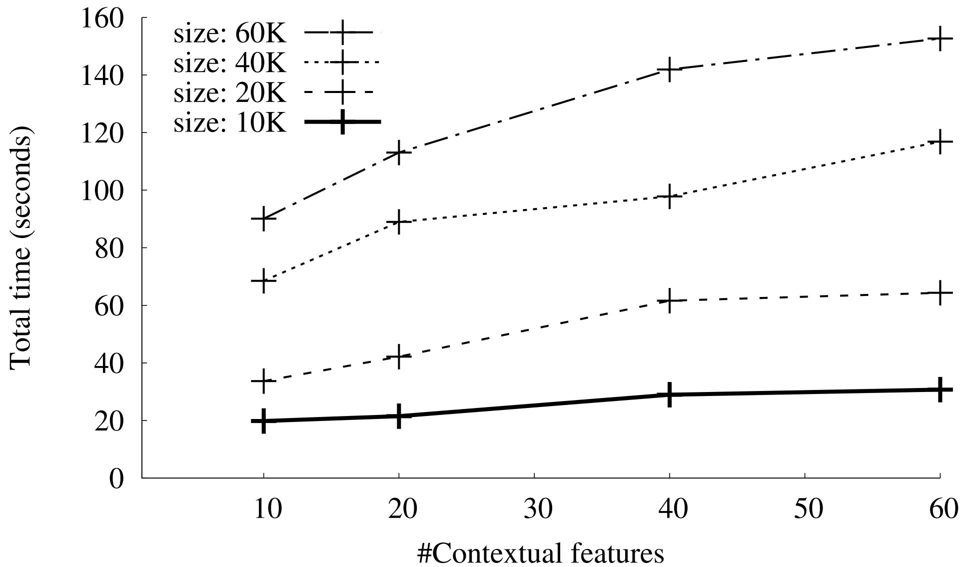
```
Report of association of O=Price on Si=Race:
Global Population of size 494436
p-value = 2.31e-178 ; NMI = [0.0241, 0.0286]
+-----+----------+----------+----------+---------------------+----------------+-----------+------------+
|Price|     Asian|     Black|  Hispanic|Indian & Alaska Native|Pacific Islander|      White|       Total|
+-----+----------+----------+----------+---------------------+----------------+-----------+------------+
|High |  430 (2%)|  977 (2%)| 4013 (5%)|             654(19%)|        55 (7%)| 22544 (7%)| 29168  (6%)|
|Low  |23244(98%)|60629(98%)|82652(95%)|            2879(81%)|       766(93%)|286877(93%)|465268 (94%)|
|Total|23674 (5%)|61606(12%)|86665(18%)|            3533 (1%)|       821 (0%)|309421(63%)|494436(100%)|
+-----+----------+----------+----------+---------------------+----------------+-----------+------------+

Subpopulation of size 7337
Context = {Gender: Female, State: NY, Income: <50K}
p-value = 4.73e-102 ; NMI = [0.0936, 0.1573]
+-----+--------+---------+---------+----------------------+----------------+-----------+----------+
|Price|   Asian|    Black| Hispanic|Indian & Alaska Native|Pacific Islander|      White|     Total|
+-----+--------+---------+---------+----------------------+----------------+-----------+----------+
|High | 12 (2%)|  17 (1%)|  31 (2%)|                5(14%)|           0(0%)|   493(15%)| 566  (8%)|
|Low  |512(98%)|1482(99%)|1795(98%)|               31(86%)|           0(0%)|  2848(85%)|6771 (92%)|
|Total|524 (7%)|1499(20%)|1826(70%)|               36 (0%)|           0(0%)|  3341(46%)|7337(100%)|
+-----+--------+---------+---------+----------------------+----------------+-----------+----------+
```

Fig. 12: **Disparate impact of a Staples-inspired 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) ***Training* Phase**



(b) ***Testing* Phase**

Fig. 13: **Performance Benchmark of FairTest Operations:** Time spent in hypothesis generation phase (a) and testing phase (b). Uses Microbenchmark as workload and averages over 10 iterations.