FairTest: A Unit Testing Framework for Uncovering Privacy Bugs in Data Driven Applications

V. Atlidakis and X. He

May 5, 2015

Abstract

Modern web applications are increasingly data-driven and assist users with their daily tasks, ranging from browsing social networks to issuing bank transactions. Users of these applications always expect a highly personalized experience. This expectation inevitably steers the collection and processing of data to profile individuals and infer their preferences. Despite the phenomenal evolution in techniques and tools to process personal data and infer users' preferences, developers notoriously miss tools to increase transparency of this data. Also, developers miss support to evaluate if policies are enforced correctly across collected data and if algorithms are creating objectionable biases within user populations. In this paper we theoretically formalize the notion of discriminatory treatment of users in modern data-driven web applications. Then, we design, implement, and evaluate *FairTest*, a tool to increase developers visibility into the implications of various data-use policies by reporting discriminatory treatment of users – or, *privacy bugs*. Our results show that *FairTest* can uncover *privacy bugs* in the pricing policy of the "Staples Inc." online store.

1 Introduction

Modern web applications are increasingly data-driven and assist users with their daily tasks, ranging from browsing social networks to issuing bank transactions. Users are accessing these applications from various personal devices such as smartphones, tablets, and desktop PCs, and always expect a highly personalized experience. Therefore, modern applications are now metamorphosing the internet from an unbiased, impersonal service into a highly personalized service that dynamically reforms to match each individual's preferences. This personalized era of the internet is based on the continuous collection and processing of data to profile individuals and infer their preferences.

Although there is a phenomenal evolution in the techniques and tools used to process collected data and infer users' preferences, there is a stagnation in the development of tools to increase transparency of this data. All the more, highly sensitive personal information is being used from applications to enhance user experience and help create efficient policies for online advertising, pricing, and hiring. Yet, developers notoriously miss tools to evaluate whether policies are enforced correctly across collected data and whether their algorithms are creating objectionable biases within user populations or not. Clearly, the unpredictable ways in which data is being used, carry the danger of unintentional data misuse, such as algorithmic discrimination of seemingly unrelated users.

An indicative motivating example of data misuse is the "Staples Inc." different treatment case, reported by the Wall Street Journal in 2012 [6]. In this case, an arguably reasonable algorithmic decision to customize online prices based on user proximity to competitor brick-and-mortar stores, effectively led to objectionable bias: Areas that tended to see discounted prices had a higher income than areas that tended to see higher prices. Other examples of differential treatment in web applications – intentional or unintentional – have been identified in a variety of other contexts, including advertising [5], pricing [4], and hiring [1]. This kind of indirect – and likely unintended – effects are challenging to identify, and with the ubiquitous use of sensitive information collected by web applications, the risk for these dangers – or, *privacy bugs* – is only increasing.

In this paper we present the theoretical foundation to formalize the notion of discriminatory treatment of users in modern web applications. Then, we build *FairTest*, a tool to increase developers visibility into the implications of various data-use policies by reporting discriminatory treatment – or, *privacy bugs*. Our theoretical foundation for identifying discriminatory treatment of users examines the correlation between a set of outputs (shown to users) and a set of user attributes. More precisely, user attributes are split into two categories: (a) a set of protected user attributes such as race or sexual orientation – which applications should not use to implement their policies and produce outputs – and (b) a set of input attributes which applications is legitimate to use to produce outputs. Based on this rule, *FairTest* reports as *privacy bugs* any strong correlations between protected user attributes and outputs shown to users.

We build *FairTest* and provide it as a service with a REpresentational State Transfer (REST) Application Program Interface (API). Clients of the *FairTest* service – let them be application developers or external auditors, e.g., the Federal Trade Commission (FTC) – use its REST API to register users and corresponding outputs. Based on a set of registered users and the corresponding outputs, *FairTest* then creates bug-reports and publishes them into its web interface. These reports indicate any strong correlations between (protected) user attributes and outputs. In Section 3 we present in detail the architecture of *FairTest*'s implementation and its API. In order to evaluate our *FairTest* prototype, we used one million synthetic users that match the demographics of US population, simulate "Staple Inc." pricing policy, and use *FairTest* to generate bug-reports. Our results include bug-reports that identify the discriminatory behavior of "Staple Inc." pricing policy. In Section 4 we detail more on our experimental methodology and on the results of our evaluation.

Overall, this paper includes three major contributions:

- The theoretical background necessary to formulate and evaluate the presence of discriminatory treatment of users of modern data-driven web applications.
- The design and implementation of *FairTest*, a service for identifying and reporting discriminatory treatment of users of modern data-driven applications.
- The evaluation of *FairTest* in the "Staples Inc." pricing engine, a well know case of discriminatory treatment of users.

The rest of this paper is organized as follows. Section 2 present our theoretical foundations along with a set of motivating examples. Section 3 gives an overview on our *FairTest* implementations and its API. Section 4 presents results of the evaluation of our *FairTest* prototype. Section 5 describes related work, and finally, section 6 includes our concluding remarks and directions for future work.

2 Motivation and Theoretical Background

In this section we motivate our work and establish the theoretical background neccessary for building *FairTest*. We start with examples that intuitivelly introduce the notion of discriminatory treatment of users of modern data-driven applications. Then, we formalize *statistical parity* and *relaxed statistical parity* to quantify the presence of discriminatory treatment of populations.

As a first, simple example consider a hypothetical online store that sells product in either "low" or "hight" prices. That is, each users sees either a "low" or a "high" price uppon visiting the web site of hypothetical store. Also, suppose that this hypothetical online store has 100 active users that are spread into three different groups A, B, and C, as shown in Table 1. Intuitivelly, users of the population A are treated unfavorably compared to users of populations of B and C. This is because users who belong to population A receive high prices more often than users who belong to the populations B and C. In other words, the probability that a user will be presented with a high price is higher if he or she belong to population A, and lower if he or she belongs to population B and C. Therefore, there is no *statistical parity* among the users of populations A, B, and C, since the hypothetical online store customizes price and disciminates based on users' membership on a population.

On the other hand, for the same hypothetical online store, consider that the users are distributed into three populations A, B, and C, as shown in Table 2. In this case users of each population see approximatelly the some proportion of high versus low prices. In other words, a user has approximately the same probability to receive a high or a low price, regardless of the population to which he or she belongs to. Trerefore, there is no population whose users are treated infavorably.

The previous two motivating examples intuitivelly leads us use *statistical parity* as a simple metric indicating discriminatory treatment of populations. In what follows, we more formally introduce the condition asserting *statistical parity* among users of populations.

2.1 Statistical Parity

For two sets of users S and S', and some output O, *statistical parity* asks that:

$$|P\{O|x \in S\} - P\{O|x \in T\}| \leq \varepsilon \qquad (1)$$

Population	#Members	Price		Statistical Parity
		Low	High	(for high price)
A	30	10	20	$0.198 = \lvert \frac{20}{30} - \frac{33}{70} \rvert$
B	30	16	14	0.090
C	40	21	19	0.091
Total	#100	47	53	-

Table 1: **Discriminatory behavior against population(s).** Users of population A receive twice as many high prices as low, while users of populations B and C receive approximately the same ammount of high and low prices. Therefore, the probability that a user will receive a high price depends on the population to which he or she belongs to, and condition 1 for *statistical parity* yields a higher delta compared to the previous example.

Population	#Members	Price		Statistical Parity
		Low	High	(for high price)
A	30	15	15	$0.028 = \lvert \frac{15}{30} - \frac{37}{70} \rvert$
B	30	14	16	0.019
C	40	19	21	0.008
Total	#100	48	52	-

Table 2: **Non-discriminatory behavior.** Users of three populations receive approximately the same proportions of low versus high prices. Therefore, the probability that a user will receive a high price is independent of the population to which he or she belongs to, and condition 1 for statistical parity yields a low delta.

Note that if users are split into more than two populations then statistical parity requires that condition 1 holds for seperately for each population against all the rest.

Having introduced a formal metric for asserting *statistical parity*, we now revisit Tables 1 and 2. In Table 1, we observe that for population A the delta of condition 1 is 0.198 for high prices. While, in Table 2, for population A the delta of condition 1 is 0.028 for high price. Observe that the value of the delta in the first case is one order of magnitude higher than the respective value in the second case. This is because in the first case users of A are receiving a high price much more often than in the second case. Hence, in the first case there is low *statistical parity* among users of populations A, B, and C; while in the second case, there is high statistical parity among users of populations A, B, and C.

Although simple and intuitive, the above definition of *statistical parity* fails in cases where developers implement policies that inherently discriminate users based on some requirement. Such cases may arise when there is a business necessity which is relevant to the policy of pricing, advertising, or hiring. For instance, consider the example of a hypothetial bank that issues two types of loans: (a) payday loans that are issued to customers without credit history and have high interest rate, and (b) personal loans that are issued to customers with credit history and have low interest. Suppose that this hypothetical bank serves 300 customers as shown in Table 3.

At first sight users of population A are being treated unfavorably because they are proportionaly taking more payday loans than users of population B and C. Specifically, users of population A receive 25% and 20% more payday loans than users of populations B and C, respectively. However, upon closer examination, one observes that only 20% of As users have credit history (which is a prerequisite for personal loans) against 55% of Bs and Cs users. Therefore, bussiness necessity requires that before examining *statistical parity*, users should be discriminated based on whether they have credit history or not. Since in presence of bussiness necessity satisfying *statistical parity* is not reasonable, in what follows we introduce a relaxed version of *statistical parity*.

2.2 Relaxing Statistical Parity

For a sets of users S and S', a bussiness necessity (requirement) R, and some output O, *relaxed statistical parity* asks that:

$$|P\{O|x \in S \cap R\} - P\{O|x \in T \cap R\}| \leq \varepsilon \qquad (2)$$

and

$$|P\{O|x \in S \cap R'\} - P\{O|x \in T \cap R'\}| \leq \varepsilon \qquad (3)$$

Essentially, conditions 2 and 3 let the users be discriminated into two sets R (users that meet the bussiness necessity) and R' (users that do not meet the business necessity). However, conditioned on being in R (or not), there is no additional discrimination between the sets S and S. This definition could easily be extended to deal with non-binary categories of utility, where business-necessity implies splitting the user base into multiple sets.

Having established the notion of *relaxed statistical parity* we now revisit the example of a hypothetical bank that issues payday and personal loans to 300 users, and compare *statistical payday* againsts *relaxed statistical parity*. The users of the hypothetical bank are again distributed into three populations, as shown in Table 3. Also, in Table 4 demonstated the *statistical parity* and the *relaxed statistical parity* for

Credit	Loan Type (Population A)			Loan Type (Population B)			Loan Type (Population C)		
history	Payday	Personal	Total	Payday	Personal	Total	Payday	Personal	Total
YES	5	15	20	15	40	55	10	45	55
NO	80	0	80	45	0	45	45	0	45
Total	85	15	100	60	40	100	55	45	100

Table 3: **Discriminatory behavior on presence of bussiness necessity (credit history).** At first sight users of population A are proportionaly taking more payday loans (payday loans come with higher interest than personal loans) than users of population B. Specifically, users of population A receive 25% more and 20% more payday loans than users of populations B and C, respectively. However, upon closer examination, one notes that only 20% of A's users have credit history (which is a prerequisite for personal loans) against 55% of B's and C's users. Therefore, bussiness necessity requires that before examining statistical parity, users should be discriminated based on whether they have credit history or not.

Credit	Loan Type (Population A)			Statistical Parity (Population A)		Relaxed Statistical Parity (Population A)	
history	Payday	Personal	Total	Payday	Personal	Payday	Personal
YES	5	15	20	-	-	0.022	0.022
NO	80	0	80	-	-	0	0
Total	85	15	100	0.275	0.70	-	-

Table 4: **Relaxing statistical parity on presence of bussiness necessity (credit history).** Without considering business necessity, i.e., credit history, condition 1 for statistical parity yields a higher delta than if we consider business necessity, let the users be discriminated on whether they have credit history or not, and apply conditions 2 and 3.

users of population A. The delta of condition 1) for *statistical parity* of population A (middle column of Table 4) is 0.27 and 0.70 for payday and personal loans respectivelly. This implies discriminatory (unfavorable) treatment of users of population A. Indeed, there is such a discriminatory behavior that can be explained from an apriory credit history requiremnt – or, bussiness necessity. Towards incorporating bussiness necessity, the right-hand side column of Table 4 presents the values for the delta of conditions 2 and 3. In this case the deltas are 0.02 (compared to 0.27 and 0.70), because users are split into two sets based on whether they qualify for a personal loan or not, and then *relaxed statistical parity* is examined seperately.

Based on the theoretical background disscussed in Section 2.1 and 2.2 we design and implement *FairTest*; A service which, at its core, uncovers – and reports as *privacy bugs* – any violations of statistical parity among users of different populations.

3 FairTest

To address the descrimination test, We designed *FairTest*, a software testing framework for web-based applications to find out potential *privacy bugs*, whose definition was just described. The framework records both the input and output of an application, and based on the correlation between the output and certain protected private data (such as gender, race, etc.), discover if any *privacy bugs* can be discovered based on the statistical parity model.

FairTest is designed to be used for designers and engineers of data-based web applications. In addition, it can also be used for auditors or government agencies to look for evidence on certain parity violation, such as in the Staples case. The framework, for generality, is interfaced with RESTful API that can easily connect to existing software solutions. We expect that developers will only make minimal modifications to the application (mostly adding the tracking code) to derive the data for audit in *FairTest*. The framework itself is implemented in Python and Django, with support to both SQLite or PostgreSQL database.

Once *FairTest* has information about the users and the corresponding outputs from the software being tested, it can generate statistical reports evaluating the statistical parity between the protected info, the actual input and the output. This info can then be reported either through a RESTful API or a web interface to the developer or tester. With the info, we can then suggest potential *privacy bugs* that are caused by the statiscal parity found by the system, offering suggestion to the developer if the customization engine appears acceptable or not.

This section will be divided into the following parts. In subsection 3.1, the *FairTest* archetecture will be discussed; and subsection 3.2 will discuss the API design of *FairTest*.

3.1 Architecture

The *FairTest* architecture contains three most important parts: the input API, the data analyzer and the result reporter. Together with an application, the architecture is shown in Figure 1.

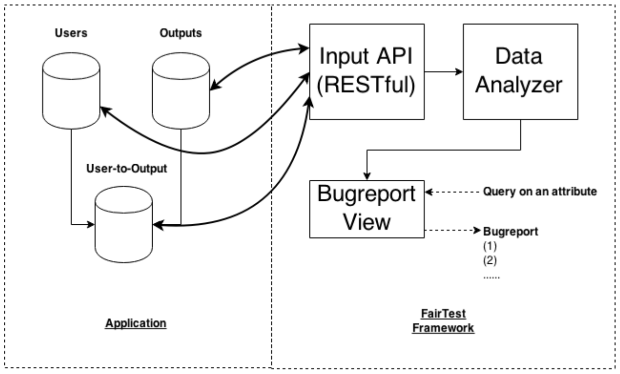


Figure 1: **FairTest Architecture**. The diagram shows a very simple application hooked up with the FairTest architecture. Bold lines are API calls to be added by the developer to the application code, and dash lines are used by the auditors to get the result. Solid line shows data flow as part of the application or testing framework.

In the figure, it shows that the input API is derectly linked with the

3.2 API

Describe maybe with a table

- Describe
- Explain why we think it is a good API? (e.g., is it extensible?)

4 Evaluation

We have implemented a *FairTest* prototype as a Django web application that features the REST API presented in Section 3. We evaluated our prototype by using it to uncover *privacy bugs* in the pricing policy of the "Staples Inc.", as described by the Wall Street Journal in 2012. We sought answer to the following three questions:

Q1 Can *FairTest* uncover *privacy bugs* for a pricing policy similar to the policy of the "Staples Inc." online store?

Q2 What is the impact of *privacy bugs* in the outputs shown to users?

Q3 What is impact of pricing policies on *privacy bugs*? Are *privacy bugs* inherent to any pricing policy, or are dependant on pricing policies?

4.1 Methodology

Our experimental evaluation of *FairTest* consists of the following three steps:

1. Generate one million synthetic users that are spread accross the areas corresponding to 40,000 US zip-codes. These synthetic users have four attributes: zip-code, race, sex, and income, and match the demographic characteristics of the US population, according to the US Census Bureau [3], as of May 3, 2014. [1]
2. With the users generated in Step 1, simulate one million visits on an online store whose pricing policy is similar to the pricing policy of the "Staples Inc.". After each visit, register to *FairTest* the respective synthetic user along with the ouput received.
3. Query *FairTest*'s web interface to uncover any strong correlation between outputs and protected user attributes, i.e., race, income, and sex.

Since user-attributes match the demographics of the US popation, race, sex, and income are assigned values that depend on zip-code. This is because the later defines the area in which a user lives in. In other words, user's race, sex, and income correlate with user's zip-code, but not amongst each other. Each synthetic user has one of the following seven races: "White", "Hispanic", "African American", "Indian or Alaskan", "Asian", "Pacific Islander", "Other", and "Two or More". Also, each user is either "Male" or "Female". Finally, each user has an income the

[1] The infrastructure necessary for generating syntetic users is provided by a project of the "Advanced Distributed Systems" course, tought by professor Roxana Geambasu, Spring 2015. The developers of this project are Z. Zhou, Z. Wan, and X. Ma.

lies into one of the following seven ranges: "less then $5,000", "more than $5,000", "more than $10,000", "more than $20,000", "more than $40,000", "more than $80,000", "more than $160,000", "more than $320,000".

4.2 Uncovering *Privacy Bugs*

In this section we use *FairTest* to uncover potential *privacy bugs* when one million synthetic users visit an online store which implements the following pricing policy: *if a user's distance from a competitor's store ("OfficeDepot & OfficeMax") is less than 20 miles, show a low price; otherwise, show a high price.* This pricing schema is similat to the pricing policy implemented by the "Staples Inc." online store, as of the Wall Street Journal, 2012.

To evaluate the presence of *privacy bugs*, we measure the value of the delta of *statistical parity* (condition 1 - introduced in Section 2.1) We group users based on (a) income, (b) race, and (c) sex, and examine potential violations of condition 1 for $\epsilon = 0.05$ and output being "high" price. Figure 2 shows the deltas for user's race, income, and sex, as a function of price engine's dependency on user' s location. When price engine's dependency on user's location is zero, all users receive random prices. When price engine's dependency on users location is 100%, all users receive prices strictly based on the rule introduced in the previous paragraph.

Figure 2(a) reveals that when users are grouped according to their income, *statistical parity* is not dependant on user's location, except for users with annual income "more than $320,000". Users whose income lie in the later categorie are treated differently (favorably or unfavorably) than the rest, and their location correlates with the outputs (i.e., prices) they receive. Also, Figure 2(b) indicates that when users are groupped according to their race, *statistical parity* is not dependant on user's location, unless the user is an Indian American or an Alaskan. This signals that Indian Americans and Alaskans are treated differently (favorably or unfavorably) than the rest.

Given the fact that both user's race and user's income are a protected attribute, one can use *FairTest*'s reports to conclude that the impelented pricing policy imposes an undesirable privacy bug, which is summarized as follows: "If the user is an Indian American or an Alaskan, or has an income more than $320,000, he or she will consistently be treated diffently than others." On the other hand, Figure 2(c) indicates when users are grouped according to their sex, *statistical parity* is independent of user's location, and therefore males are treated similarly to females.

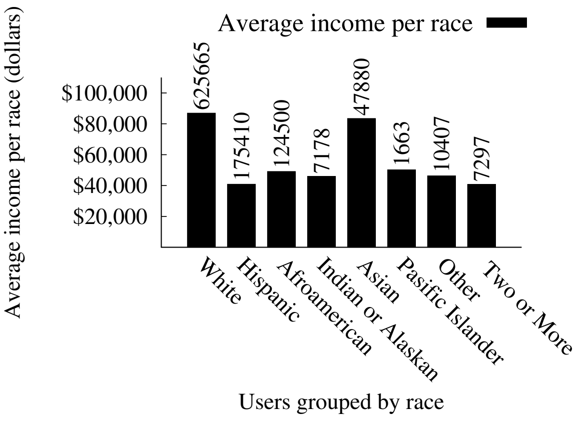


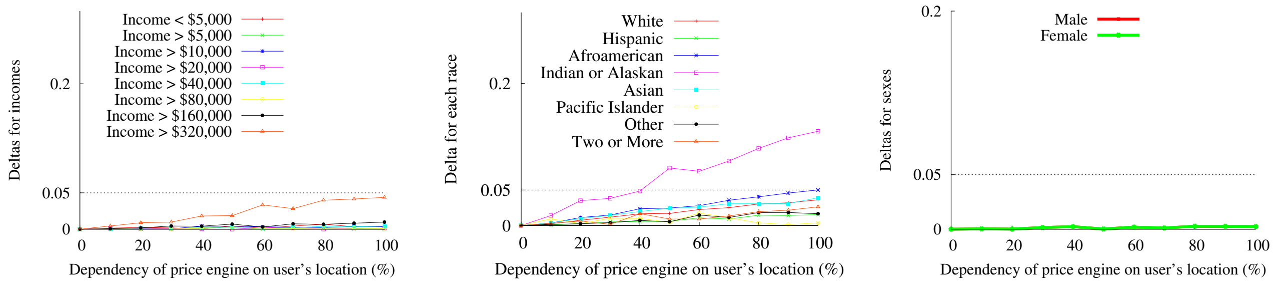
Figure 4: **Average annual income per race.** An Indian or an Alaskan user has a considerably lower income than a white American, and yet, as shown in Figure 3(a), he or she receives proportionally more high prices.

4.3 Measuring the Impact of *Privacy Bugs*

Having concluded that the aforementioned pricing policy violates *statistical parity* of users, i.e., introduces *privacy bugs*, we now investigate the impact of these bugs. To this end, we examine whether users are treated favorably or unfavorably by measuring the proportion of "high" vesus "low" prices shown to users. Figure 3 shows, in percentage, the proportion of "high" vesus "low" prices shown to them grouped by user's (a) race, (b) income, and (c) sex. Also, on top of each bar the raw number of members of the respective group is shown.

Figure 3(a) shows that for each income group, around 6% of users receive "high" prices (the rest receive "low" prices), except the income group which includes users with annual income "more than $320,000". Recall that based on Figure 2(a), we had already concluded that users with annual income "more than $320,000" are treated differently that the rest. Additionally, Figure 3(a) helps conclude that users with annual income "more than $320,000" are treated favorably compare to the rest. Note that a user with annual income "less than $5,000" is more probable to receive a high price, than a user with annual income "more than $320,000". Clearly this is a very poor – yet unintentional – algorithmic decision imposed by a privacy bug which is difficult to foresee without the aid of *FairTest*.

Figure 3(b) helps conclude that the percentage of "high" prices shown to "Indian Americans and Alaskans" is notably higher than any other race. Recall that by analyzing Figure 2(b), we had already



(a) Income (b) Race (c) Sex

Figure 2: **Statistical parity and its dependency on user's location.** Shows the dependency of statistical parity on user's location, when users are grouped based on (a) income, (b) race, and (c) sex. Figures (a) and (b) reveal that when users are grouped according to income and race, statistical parity correlates for some groups with the dependency of the price engine on user's location. While Figure (c) reveals that statistical parity is independent of user's location when users are grouped based on sex.

concluded that "Indian Americans and Alaskans" are treated diffently than the rest. In addition, Figure 3(b) reveals that "Indian Americans and Alaskans" are treated unfavorably, since they are more probable to receive a "high" price, than the rest. This is also a very poor algorithmic decision, because as shown in Figure 4, "Indian Americans and Alaskans" have a low annual income on average. In other words, the enforced pricing policy consistently shows more "high" prices to populations with lower annual income. Although this is an unintentional effect of a reasonable algorithmic decision, it may actually attract negative critisism. We believe that *FairTest* helps increase developers' transparency on such kind of side effects, and minimize the unpredictable danger of discriminatory treatment of populations.

4.4 Measuring the Impact of Pricing Policies

Finally, we examine the impact of different pricing policies on the presence of *privacy bugs*, and seek to understand how algorithmic decision affect *privacy bugs*. To achieve this, we modify the pricing policy to now abide by the following rule: *if a user's distance from a competitor's store ("OfficeDepot & OfficeMax") is less than 10 miles, show a low price; otherwise, show a high price.* Then, we compare the outputs shown to users in this case, against the outputs shown to users with the policy introduced in Section 4.2. Note that the only modification is the change on user's distance from competitor's stores (from 20 miles to 10 miles).

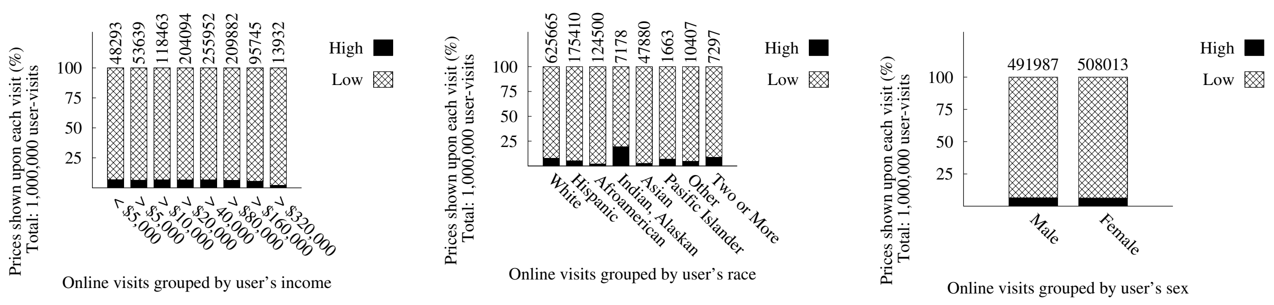
Figure 5 shows that after applying the new pricing policy more users receive "high" prices compared to Figure 3. This is expected since we apply a stricter policy that shows "low" if the competitor's stores are within 10 miles, instead of 20 miles. Notably, however, Indian Americans and Alaskans are still the most unfavorably treated populations. Also, users with annual income "more than $320,000" are still the most favorably treated poulation. This indicates that despite variations of the pricing policies, there are inherent correlations between user attributes that are transparent to developers, have it not been *FairTest*. Hence, we believe that *FairTest* helps increase developers transparency on data use and provide some understanding on: the policies applyied on the data.

4.5 Summary of Results

Overall, these result highlight the danger of discriminatory treatment of users as a results of legitimate algorithmic decisions. Although developers design policies to optimize for their needs, these policies have unpredictable side effects on data, and may result into objectionable situations. With simulating "Staples Inc." pricing policy, we show-cased how *FairTest* helps increase developers' transparency on data use, uncover *privacy bugs*, and report these bugs to developers.

5 Related Work

Fairness in the use of data by web applications is drawing increasing attention in our days, and yet there is no established methodology for measuring fairness of online algorithms producing recommendations, nor is there an established methodology for shading some light and helping uncover potential on-



(a) Income (b) Race (c) Sex

Figure 3: **Prices shown to users and their dependency on income, race, and sex.** Shows the proportion of high versus low prices shown to users based on (a) income, (b) race, and (c) sex. Figure (a) reveals that a user with annual income less than $5,000 receives proportionaly more high prices than a user with annual income more than $320,000. Figure (b) indicates that an Indian American or an Alaskan users receives notably more high prices than any other user. This raises a consern, since as shown in Figure 4, an Indian American or an Alaskan user has on average a considerably lower annual income than a white American user. Figure (c) shows that male and female users receive approximately the same proportion of high versus low prices.

line discrimination cases. The authors of [2] study fairness of classification and their purpose is to prevent discriminating individuals based on certain characteristics, such as membership in a group, and at the same time maintain the effectiveness of the classifier. The limitation of their approach is that it requires programmes to be able to define similarity metrics, and it is unclear whether programmers are able to define such metrics or not. Also, citeDisparateImpact unequal treatment of different populations by a classification methodology is studied for its legal concept of disparate impact. This work formalizes what it means for a dataset to have the potential for disparate impact if used by a classifier, and suggests methods for detecting and removing potential disparate impact. However, it is not showing the effectiveness of the proposed procedures in real applications, and also does not extend to general discrimination hazards.

6 Conclusions & Future Work

We have theoretically formalized the notion of discriminatory treatment of users in modern data-driven web applications. Also, we have designed, implemented, and evaluated *FairTest*, a tool to increase developers visibility into the implications of various data-use policies by reporting discriminatory treatment of users – or, *privacy bugs*. Our results showed that *FairTest* can uncover and measure the impact of *privacy bugs* in pricing policies similar to the policy of "Staples Inc." online store.

In the future we plan to explore the potential of using *FairTest* to evaluate the presence of *privacy bugs* on real applications – such as applications used for online recommendations and advertising. Also, we plan to evaluate *FairTest*'s effectiveness in uncovering *privacy bugs* with the use of ranking heuristics, other than *statistical parity.*

References

[1] A. Acquisti and C. M. Fong. An experiment in hiring discrimination via online social networks. *Social Science Research Network Working Paper Series*, 2012.

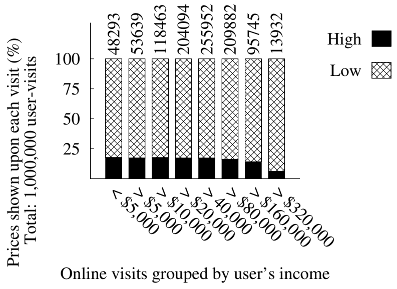
[2] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness Through Awareness. In *Innovations in Theoretical Computer Science*, 2011.

[3] Census bureau. United States Census Bureau. `http://www.census.gov/`. Accessed: 05/3/2015.

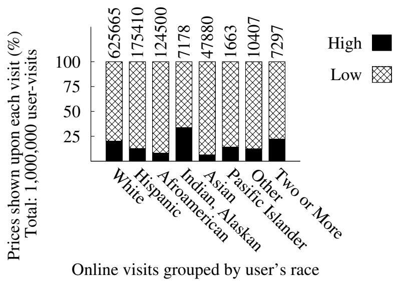
[4] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring price discrimination and steering on e-commerce web sites. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 305–318, New York, NY, USA, 2014. ACM.

[5] L. Sweeney. Discrimination in online ad delivery. *Commun. ACM*, 56(5):44–54, May 2013.

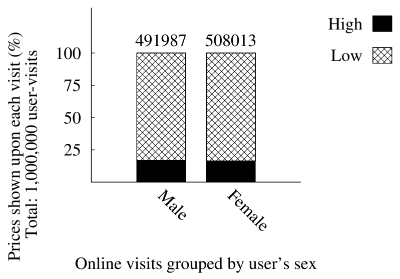
[6] THE WALL STREET JOURNAL. Websites Vary Prices, Deals Based on Users' Information. `http://www.wsj.com/articles/SB10001424127887323777204578189391813881534`. Accessed: 02/02/2015.



(a) Income



(b) Race



(c) Sex

Figure 5: **Prices shown to users and their dependency on income, race, and sex.** Shows the proportion of high versus low prices shown to users based on (a) income, (b) race, and (c) sex. In this case, the pricing policy is stricter and results to "low" prices if user's distance from a competitor's store is less than 10 miles. Figure (a) reveals that a users with annual income less than \$5,000 still receives proportionaly more high prices than a user with annual income more than \$320,000. Figure (b) indicates that an Indian American or an Alaskan users still receives notably more high prices than any other user. Figure (c) shows that male and female users receive approximately the same proportion of high versus low prices.