Themis: Automatically Testing Software for Discrimination

Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou
University of Massachusetts Amherst
Amherst, Massachusetts, USA
{rangell, bjohnson, brun, ameli}@cs.umass.edu

ABSTRACT

Bias in decisions made by modern software is becoming a common and serious problem. We present Themis, an automated test suite generator to measure two types of discrimination, including causal relationships between sensitive inputs and program behavior. We explain how Themis can measure discrimination and aid its debugging, describe a set of optimizations Themis uses to reduce test suite size, and demonstrate Themis' effectiveness on open-source software. Themis is open-source and all our evaluation data are available at http://fairness.cs.umass.edu/. See a video of Themis in action: https://youtu.be/brB8wkaUesY

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**;

KEYWORDS

Software fairness, discrimination testing, fairness testing, software bias, testing, Themis, automated test generation

ACM Reference Format:
Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2018. Themis: Automatically Testing Software for Discrimination. In *Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '18), November 4–9, 2018, Lake Buena Vista, FL, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3236024.3264590

1 INTRODUCTION

Software plays an important role in making decisions that shape our society. Software decides what products we are led to buy [36]; who gets financial loans [43]; what a self-driving car does, which may lead to property damage or human injury [24], how medical patients are diagnosed and treated [48], and who gets bail and which criminal sentence [4]. Unfortunately, there are countless examples of bias in software. Translation engines inject societal biases, e.g., "She is a doctor" translated into Turkish and back into English becomes "He is a doctor" [19]. YouTube is more accurate when automatically generating closed captions for videos with male than female voices [50]. Facial recognition systems often underperform on female and black faces [32]. In 2016, Amazon software decided not to offer same-day delivery to predominantly minority neighborhoods [35]. And the software US courts use to assess the risk of a criminal repeating a crime exhibits racial bias [4].

Bias in software can come from learning from biased data, implementation bugs, design decisions, unexpected component interactions, or societal phenomena. Thus, software discrimination is a challenging problem and addressing it is integral to the entire software development cycle, from requirements elicitation, to architectural design, to testing, verification, and validation [14].

Even defining what it means for software to discriminate is not straightforward. Many definitions of algorithmic discrimination have emerged, including the correlation or mutual information between inputs and outputs [52], discrepancies in the fractions of inputs that produce a given output [17, 26, 56, 58] (known as group discrimination [23]), or discrepancies in output probability distributions [34]. These definitions do not capture causality and can miss some forms of discrimination.

To address this, our recent work developed a new measure called *causal discrimination* and described a technique for automated fairness test generation [23]. This tool demonstration paper implements that technique for the group and causal definitions of discrimination in a tool called Themis v2.0 (building on an early prototype [23]). This paper focuses on the tool's architecture, test suite generation workflow, and efficiency optimizations (Section 2), and its user interface (Section 3). Section 4 places Themis in the context of related research and Section 5 summarizes our contributions.

2 THEMIS: AUTOMATED FAIRNESS TEST GENERATION

Figure 1 describes the Themis architecture and fairness test-suite generation workflow. Themis consists of four major components: input generator, cache, error-bound confidence calculator, and discrimination score calculator. Themis uses the input schema of the system under test to generate test suites for group or causal discrimination. Themis generates values for non-sensitive attributes uniformly randomly, and then iterates over the values for sensitive attributes. This process samples equivalence classes of test inputs. Themis later uses the system's under test behavior on these sampled subsets of the equivalence classes to compute the discrimination score. Using a cache to ensure no test is executed multiple times, Themis executes the system under test on the generated tests. Themis iterates this test generation process, generating more tests within the equivalence classes, until the confidence in the error bound satisfies the user-specified threshold, and then outputs the discrimination score. The final confidence bound is within the specified threshold, though Themis can also produce its exact measure.

Themis focuses on two measures of discrimination, group and causal. To help explain the two measures, consider a simple LOAN

ESEC/FSE '18, November 4–9, 2018, Lake Buena Vista, FL, USA

ACM ISBN 978-1-4503-5573-5/18/11…$15.00
https://doi.org/10.1145/3236024.3264590

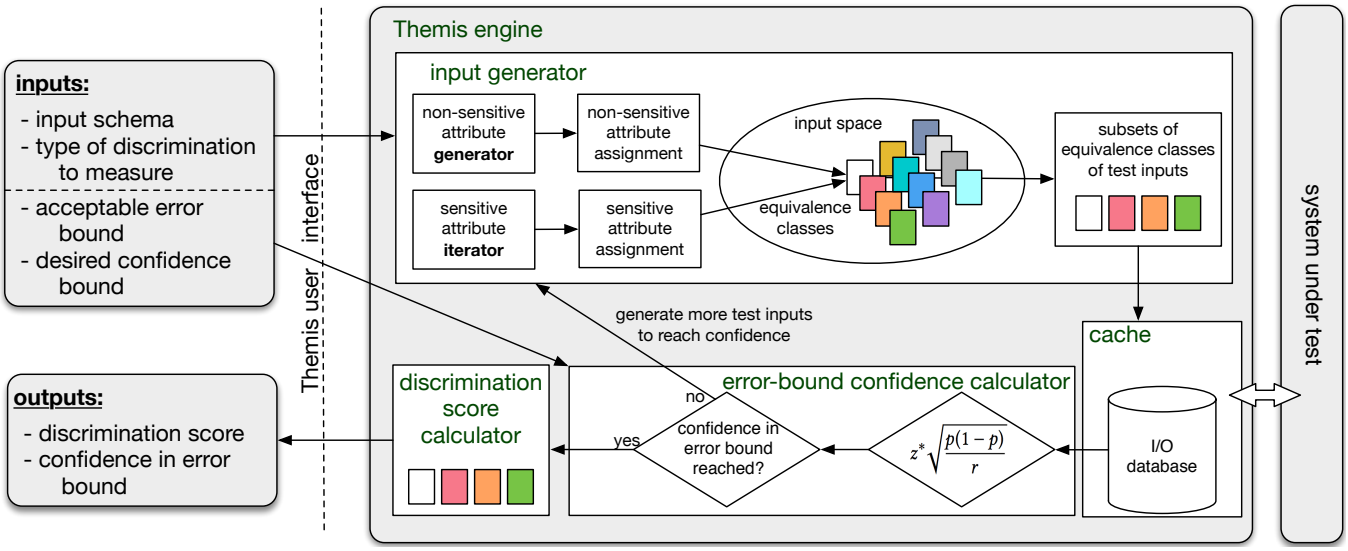


Figure 1: The Themis architecture and fairness test-suite generation workflow.

program that decides if loan applicants should be given loans. LOAN inputs are each applicant's name, age bracket (≤40 or >40), race (green, purple), income bracket, savings, employment status, and requested loan amount; the output is "approve" or "deny".

Group discrimination is the maximum difference in the fractions of software outputs for each sensitive input group. For example, LOAN's group discrimination with respect to race compares the fractions of green and purple applicants who get loans. If 35% of green and 20% of purple applicants get loans, then LOAN's group discrimination with respect to race is 35% − 20% = 15%. With more than two races, the measure would be the difference between the largest and smallest fractions. Group discrimination with respect to multiple input attributes compares the crossproduct of the attributes, e.g., for race and age bracket, there are four groups: [purple, ≤40], [purple, >40], [green, ≤40], and [green, >40].

Software testing enables a unique opportunity to conduct *hypothesis testing* to determine statistical causation [45] between inputs and outputs. It is possible to execute LOAN on two individuals identical in every way except race to verify if the race causes an output change. *Causal discrimination* is the frequency with which equivalences classes of inputs (recall Figure 1) contain at least two inputs on which the software under test produces different outputs. For causal discrimination, each equivalence class contains inputs with identical non-sensitive attribute values but varied sensitive attribute values. For example, LOAN's causal discrimination with respect to age and race is the fraction of equivalence classes that contain a pair of individuals with identical name, income, savings, employment status, and requested loan amount, but different race or age, for which LOAN approves a loan for one but not the other.

Exhaustively testing software can measure its group and causal discrimination, but it is infeasible in practice. The rest of this section describes three optimizations (previously proved sound [23]) Themis uses to reduce test suite size. Applying these optimizations to real-world software (see Section 3), reduced test suite sizes by, on average, 2,849 times for group discrimination and 148 times for causal discrimination [23]. The more software discriminates, the greater the reduction in test suite size.

Sound pruning. The number of possible executions grows exponentially with the number of input attributes being tested for discrimination. However, the group and causal discrimination definitions are monotonic: if software discriminates over threshold θ with respect to a set of attributes X, then the software also discriminates over θ with respect to all supersets of X (see Theorems 4.1 and 4.2 and their proofs in [23]). This allows Themis to prune its test input space. Once Themis discovers that software discriminates against X, it can prune testing all supersets of X.

Further, causal discrimination always exceeds group discrimination with respect to the same set of attributes (see Theorem 4.3 and its proof in [23]) so Themis can prune its test input space when measuring both kinds of discrimination: If software group discriminates with respect to a set of attributes, it must causally discriminate with respect to that set at least as much.

These observations and their formal proofs allow Themis to employ a provably sound pruning strategy (Algorithm 3 in [23]).

Adaptive sampling. Themis approximates group and causal discrimination scores through sampling done via iterative test generation (recall Figure 1). Sampling in Themis is adaptive, using the ongoing score computation to determine if a specified bound of error with a desired confidence level has been reached. Themis generates inputs uniformly at random using an input schema, and maintains the proportion of samples evidencing discrimination, computing the bound of error for that proportion.

Test caching. Themis may generate repetitive tests: tests relevant to group discrimination are also relevant to causal discrimination, and tests relevant to one set of attributes can also be relevant to another set. This redundancy in fairness testing allows Themis to exploit caching to reuse test results without re-executing tests.

3 USING THEMIS TO DISCOVER AND DEBUG DISCRIMINATION

Themis is a standalone application written in Python. Themis is open-source: http://fairness.cs.umass.edu/. This paper describes Themis version 2.0. This section uses a simple example LOAN implementation (Figure 2a) to demonstrate Themis.

Themis automatically generates tests to detect and measure group and causal discrimination. Themis' inputs are a path to the executable for the software to be tested, an input schema, and what to measure.

To test LOAN for discrimination, the user specifies the LOAN input schema either via Themis' GUI (Figure 2b), or via a configuration file. The input scema includes every input attribute to the software to be tested and the range of values it can take on, e.g., sex → {male, female}. Currently, Themis handles categorical inputs, such as race, income brackets, etc. To specify what to measure, the user selects group or causal discrimination (or both), a set of input attributes with respect to which to test, an acceptable error bound, the desired confidence in that bound, and a maximum acceptable discrimination threshold. Themis allows saving (and loading) these settings for later use.

Themis generates and executes a test suite to perform the specified measurements. If it finds discrimination, the user can choose to explore that discrimination further. Themis displays the details of observed group and causal discrimination differently. Figure 2c shows the group discrimination details Themis found for LOAN when asked to measure discrimination with respect to sex, race, and income. LOAN group discriminates with respect to income 50.4% of the time (above the 20.0% threshold): LOAN approved 49.6% of loan applicants who make between $50,000 and $100,000, and 100% of applicants who make more than $100,000. As this example illustrates, not all discrimination may be undesirable. Approving loans based on income may very well be a desirable behavior. It is up to the Themis user to decide which input attributes need to be tested for discrimination.

Figure 2d shows the causal discrimination details Themis found for LOAN. In addition to the discrimination similar to the group measurements for income, Themis also found 66.9% causal discrimination with respect to race. Themis lists the tests that exhibit the causal discrimination. For example, LOAN approves a loan for a male, orange candidate with income between $50,000 and $100,000, but denies a loan for a blue candidate who is otherwise identical. These causal pairs allow investigating the source of discrimination by tracing through the executions on two (or more) relevant inputs that exhibit the different behavior, and to potentially debug the problem.

As Themis implements an existing automated fairness test suite generation technique [23], we now briefly summarize the earlier evaluation of that technique. Evaluated on a benchmark of eight open-source software systems and two financial datasets (available at http://fairness.cs.umass.edu/), automated fairness test suite generation was effective at discovering both group and causal discrimination. Causal discrimination sometimes captured bias that group discrimination missed. For example, one of the discrimination-aware decision-tree implementations [26] trained not to discriminate with respect to gender exhibited causal gender discrimination of 11.3%. Learning-based systems can find ways to discriminate

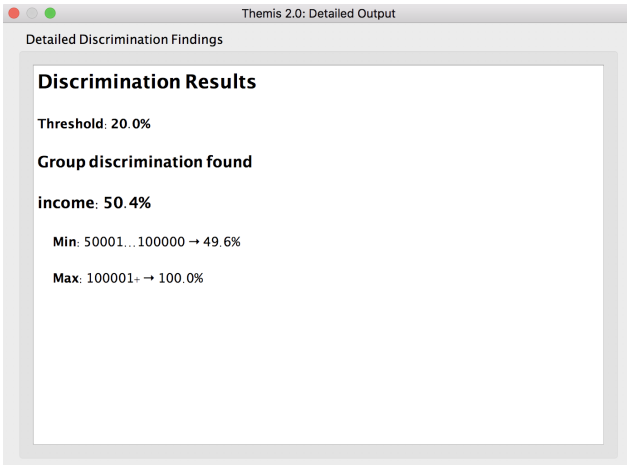
```
1  if (race is green or orange)
2    if (income > $50, 000)
3      approve
4    else
5      deny
6  else
7    if ($50, 000 < income ≤ $100, 000)
8      deny
9    else
10     approve
```

(a) Sample LOAN implementation.

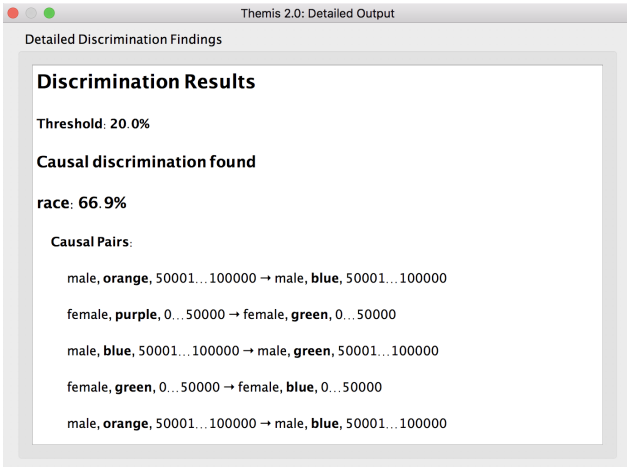
		Input Name	Input Type	Possible Values
Delete	Edit...	income	Categorical	{0...50000,50001...100000,100001+}
Delete	Edit...	race	Categorical	{blue,green,orange,purple}
Delete	Edit...	sex	Categorical	{male,female}

Add Input

(b) LOAN input schema.



(c) Themis view of group discrimination results.



(d) Themis view of causal discrimination results.

Figure 2: Themis tests a LOAN implementation (a) using an input schema (b) and finds group (c) and causal (d) discrimination.

against certain individuals in one way, and certain other individuals in another way such that those two discrimination instances cancel out with respect to group discrimination. Causal discrimination, however, properly captures such bias. Sometimes, avoiding discrimination against one attribute may increase discrimination against another. For example, the evaluation showed that training not to discriminate with respect to gender can lead to a significant increase in discrimination against race. Forcing additional constraints on machine learning may have unexpected consequences, making Themis' discrimination testing even more important. Themis' optimizations (recall Section 2) reduced test suite sizes by, on average, 2,849 times for group discrimination and 148 times for causal discrimination [23].

4 RELATED WORK

Discrimination shows up in many software applications, e.g., advertisements [49], hotel bookings [36], and image search [29]. Meanwhile, software is entering domains in which discrimination could result in serious negative consequences, including criminal justice [4], finance [43], and hiring [46]. Software discrimination may occur unintentionally, e.g., as a result of implementation bugs, as an unintended property of self-organizing systems [5, 10, 12, 13], as an emergent property of component interaction [6, 11, 16, 33], as conflicting logic from multiple developers' changes [7–9, 15] or as an automatically learned property from biased data [17, 18, 25–28, 56–58].

Themis focuses on two measures of discrimination, group and causal. Group discrimination is a generalization of the Calders-Verwer (CV) score [18], used frequently in prior work on algorithmic fairness, particularly in the context of fair machine learning [17, 26, 56, 58]. Many other definitions exist. The group discrimination definition can be generalized to rich subgroups [30, 30]. Another defintion defines discrimination by observing that a "better" input is never deprived of the "better" output [20]. That definition requires a domain expert to create a distance function for comparing inputs. Causal discrimination [23] (which Themis measures) goes beyond prior work by measuring causality [45]. Fairness in machine learning research is similarly moving toward causal measures of discrimination, e.g., counterfactual fairness [34]. FairML [1] uses orthogonal projection to co-perturb attributes, which can mask some discrimination, but find discrimination that is more likely to be observed in real-world scenarios.

FairTest [52] uses manually written tests to measure four kinds of discrimination scores: the CV score and a related ratio, mutual information, Pearson correlation, and a regression between the output and sensitive inputs. By contrast, Themis generates tests automatically and also measures causal discrimination.

Reducing discrimination in machine learning classifiers [2, 17, 26, 30, 31, 51, 56, 58] and selection algorithms [47] is important work that is complementary to ours. We focus on measuring discrimination via software testing, not developing methods for removing it. Themis can be used to manually debug discrimination bugs and thus remove discrimination. Work on formal verification of non-discrimination [3] is similarly complementary to our testing approach.

Causal relationships in data management systems [22, 37, 38] can help explain query results [40] and debug errors [53–55] by tracking and using data provenance [39]. For software systems that use data management, such provenance-based reasoning may aid testing for causal relationships between input attributes and outputs. Our prior work on testing software that relies on data management systems has focused on data errors [41, 42], whereas this work focuses on testing fairness.

Unlike other automated test generation tools, e.g., Randoop [44] and EvoSuite [21], Themis processes test results to compute discrimination scores. Prior tools are not designed for measuring discrimination, instead satisfying testing goals such as maximizing coverage. This leads to diverse test suites [21, 44]. By contrast, e.g., to measure causal discrimination, Themis has to generate pairs of similar, not diverse, inputs unlikely to be produced by other tools.

5 CONTRIBUTIONS

We have demonstrated Themis, an open-source implementation of an automated test generation technique [23] for testing software for causal and group discrimination. Themis employs three optimizations, which significantly reduce test suite size. Such test suite generation is effective at finding discrimination in open-source software. Overall, Themis is the first automated test generator for discrimination testing and serves both as a useful tool for practitioners and a baseline for future research.

ACKNOWLEDGMENT

This work is supported by the National Science Foundation under grants no. CCF-1453474, IIS-1453543, CNS-1744471, and CCF-1763423.

REFERENCES

[1] Julius Adebayo and Lalana Kagal. Iterative orthogonal feature projection for diagnosing bias in black-box models. *CoRR*, abs/1611.04967, 2016.
[2] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018.
[3] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya Nori. FairSquare: Probabilistic verification for program fairness. In *ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA)*, 2017.
[4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica*, May 23, 2016. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.
[5] Yuriy Brun, Ron Desmarais, Kurt Geihs, Marin Litoiu, Antonia Lopes, Mary Shaw, and Mike Smit. A design space for adaptive systems. In Rogério de Lemos, Holger Giese, Hausi A. Müller, and Mary Shaw, editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475, pages 33–50. Springer-Verlag, 2013.
[6] Yuriy Brun, George Edwards, Jae young Bang, and Nenad Medvidovic. Smart redundancy for distributed computation. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 665–676, Minneapolis, MN, USA, June 2011.
[7] Yuriy Brun, Reid Holmes, Michael D. Ernst, and David Notkin. Crystal: Precise and unobtrusive conflict warnings. In *European Software Engineering Conference and ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE) Tool Demonstrations track*, pages 444–447, Szeged, Hungary, September 2011.
[8] Yuriy Brun, Reid Holmes, Michael D. Ernst, and David Notkin. Proactive detection of collaboration conflicts. In *European Software Engineering Conference and ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE)*, pages 168–178, Szeged, Hungary, September 2011.
[9] Yuriy Brun, Reid Holmes, Michael D. Ernst, and David Notkin. Early detection of collaboration conflicts and risks. *IEEE Transactions on Software Engineering (TSE)*, 39(10):1358–1375, October 2013.
[10] Yuriy Brun and Nenad Medvidovic. An architectural style for solving computationally intensive problems on large networks. In *Software Engineering for*

Adaptive and Self-Managing Systems (SEAMS), Minneapolis, MN, USA, May 2007.

[11] Yuriy Brun and Nenad Medvidovic. Fault and adversary tolerance as an emergent property of distributed systems' software architectures. In *International Workshop on Engineering Fault Tolerant Systems (EFTS)*, pages 38–43, Dubrovnik, Croatia, September 2007.

[12] Yuriy Brun and Nenad Medvidovic. Keeping data private while computing in the cloud. In *International Conference on Cloud Computing (CLOUD)*, pages 285–294, Honolulu, HI, USA, June 2012.

[13] Yuriy Brun and Nenad Medvidovic. Entrusting private computation and data to untrusted networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 10(4):225–238, July/August 2013.

[14] Yuriy Brun and Alexandra Meliou. Software fairness. In *Proceedings of the New Ideas and Emerging Results Track at the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Lake Buena Vista, FL, USA, November 2018.

[15] Yuriy Brun, Kıvanç Muşlu, Reid Holmes, Michael D. Ernst, and David Notkin. Predicting development trajectories to prevent collaboration conflicts. In *the Future of Collaborative Software Development (FCSD)*, Seattle, WA, USA, February 2012.

[16] Yuriy Brun, Jae young Bang, George Edwards, and Nenad Medvidovic. Self-adapting reliability in distributed software systems. *IEEE Transactions on Software Engineering (TSE)*, 41(8):764–780, August 2015.

[17] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building classifiers with independency constraints. In *IEEE International Conference on Data Mining (ICDM) Workshops*, pages 13–18, December 2009.

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

[19] Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

[20] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 214–226, August 2012.

[21] Gordon Fraser and Andrea Arcuri. Whole test suite generation. *IEEE Transactions on Software Engineering (TSE)*, 39(2):276–291, February 2013.

[22] Cibele Freire, Wolfgang Gatterbauer, Neil Immerman, and Alexandra Meliou. A characterization of the complexity of resilience and responsibility for self-join-free conjunctive queries. *Proceedings of the VLDB Endowment (PVLDB)*, 9(3):180–191, 2015.

[23] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. Fairness testing: Testing software for discrimination. In *Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 498–510, September 2017.

[24] Noah J. Goodall. Can you program ethics into a self-driving car? *IEEE Spectrum*, 53(6):28–58, June 2016.

[25] Faisal Kamiran and Toon Calders. Classifying without discriminating. In *International Conference on Computer, Control, and Communication (IC4)*, pages 1–6, February 2009.

[26] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. Discrimination aware decision tree learning. In *International Conference on Data Mining (ICDM)*, pages 869–874, December 2010.

[27] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *International Conference on Data Mining (ICDM)*, pages 924–929, December 2012.

[28] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 35–50, September 2012.

[29] Matthew Kay, Cynthia Matuszek, and Sean A. Munson. Unequal representation and gender stereotypes in image search results for occupations. In *Conference on Human Factors in Computing Systems (CHI)*, pages 3819–3828, 2015.

[30] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. An empirical study of rich subgroup fairness for machine learning. *CoRR*, abs/1808.08166, 2018.

[31] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018.

[32] Brendan F. Klare, Mark J. Burge, Joshua C. Klontz, Richard W. Vorder Bruegge, and Anil K. Jain. Face recognition performance: Role of demographic information. *IEEE Transactions on Information Forensics and Security (TIFS)*, 7(6):1789–1801, December 2012.

[33] Ivo Krka, Yuriy Brun, George Edwards, and Nenad Medvidovic. Synthesizing partial component-level behavior models from system specifications. In *European Software Engineering Conference and ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE)*, pages 305–314, Amsterdam, The Netherlands, August 2009.

[34] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Annual Conference on Neural Information Processing Systems (NIPS)*, December 2017.

[35] Rafi Letzter. Amazon just showed us that 'unbiased' algorithms can be inadvertently racist. *TECH Insider*, April 21, 2016. http://www.techinsider.io/how-algorithms-can-be-racist-2016-4.

[36] Dana Mattioli. On Orbitz, Mac users steered to pricier hotels. *The Wall Street Journal*, August 23, 2012. http://www.wsj.com/articles/SB10001424052702304458604577488822667325882.

[37] Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y. Halpern, Christoph Koch, Katherine F. Moore, and Dan Suciu. Causality in databases. *IEEE Data Engineering Bulletin*, 33(3):59–67, 2010.

[38] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. The complexity of causality and responsibility for query answers and non-answers. *Proceedings of the VLDB Endowment (PVLDB)*, 4(1):34–45, 2010.

[39] Alexandra Meliou, Wolfgang Gatterbauer, and Dan Suciu. Bringing provenance to its full potential using causal reasoning. In *3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, June 2011.

[40] Alexandra Meliou, Sudeepa Roy, and Dan Suciu. Causality and explanations in databases. *Proceedings of the VLDB Endowment (PVLDB) tutorial*, 7(13):1715–1716, 2014.

[41] Kıvanç Muşlu, Yuriy Brun, and Alexandra Meliou. Data debugging with continuous testing. In *Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE) NIER track*, pages 631–634, August 2013.

[42] Kıvanç Muşlu, Yuriy Brun, and Alexandra Meliou. Preventing data errors with continuous testing. In *ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pages 373–384, July 2015.

[43] Parmy Olson. The algorithm that beats your bank manager. *CNN Money*, March 15, 2011. http://www.forbes.com/sites/parmyolson/2011/03/15/the-algorithm-that-beats-your-bank-manager/#cd84e4f77ca8.

[44] Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Ernst, and Thomas Ball. Feedback-directed random test generation. In *ACM/IEEE International Conference on Software Engineering (ICSE)*, pages 75–84, May 2007.

[45] Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

[46] Aarti Shahani. Now algorithms are deciding whom to hire, based on voice. *NPR All Things Considered*, March 2015.

[47] Julia Stoyanovich, Ke Yang, and HV Jagadish. Online set selection with fairness and diversity constraints. In *International Conference on Extending Database Technology (EDBT)*, pages 241–252, Vienna, Austria, 2018.

[48] Eliza Strickland. Doc bot preps for the O.R. *IEEE Spectrum*, 53(6):32–60, June 2016.

[49] Latanya Sweeney. Discrimination in online ad delivery. *Communications of the ACM (CACM)*, 56(5):44–54, May 2013.

[50] Rachael Tatman. Gender and dialect bias in YouTube's automatic captions. In *Workshop on Ethics in Natural Language Processing*, 2017.

[51] Philip S. Thomas, Bruno Castro da Silva abd Andrew G. Barto, and Emma Brunskill. On ensuring that intelligent machines are well-behaved. *CoRR*, abs/1708.05448, 2017.

[52] Florian Tramer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. FairTest: Discovering unwarranted associations in data-driven applications. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, April 2017.

[53] Xiaolan Wang, Xin Luna Dong, and Alexandra Meliou. Data X-Ray: A diagnostic tool for data errors. In *International Conference on Management of Data (SIGMOD)*, 2015.

[54] Xiaolan Wang, Alexandra Meliou, and Eugene Wu. QFix: Demonstrating error diagnosis in query histories. In *International Conference on Management of Data (SIGMOD)*, pages 2177–2180, 2016. (demonstration paper).

[55] Xiaolan Wang, Alexandra Meliou, and Eugene Wu. QFix: Diagnosing errors through query histories. In *International Conference on Management of Data (SIGMOD)*, 2017.

[56] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Learning fair classifiers. *CoRR*, abs/1507.05259, 2015.

[57] Richard Zemel, Yu (Ledell) Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning (ICML), published in JMLR W&CP: 28(3):325–333)*, June 2013.

[58] Indrė Žliobaitė, Faisal Kamiran, and Toon Calders. Handling conditional discrimination. In *International Conference on Data Mining (ICDM)*, pages 992–1001, December 2011.