# Influence-directed Explanations for Machine Learning

Shayak Sen

CMU-CS-18-107

April 20, 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
Anupam Datta, Chair
Jaime Carbonell
Matt Fredrikson
Sriram K. Rajamani
Jeannette M. Wing

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.*



This research was sponsored by the National Science Foundation under grant numbers CNS-1064688 and CNS-1704845, the Air Force Research Laboratory under grant numbers FA9550-12-1-0040, FA9550-17-1-0600, and FA9750-16-2-0287, the Ed and Martha Clarke Fellowship, and the Cylab Presidential Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

## Abstract

Increasingly, decisions and actions affecting people's lives are determined by automated systems processing personal data. Excitement about these systems has been accompanied by serious concerns about their opacity and the threats that they pose to privacy, fairness, and other values. Recognizing these concerns, it is important to make real-world automated decision-making systems accountable for privacy and fairness by enabling them to detect and explain violations of these values. System maintainers may leverage such accounts to repair systems to avoid future violations with minimal impact on the utility goals.

In this dissertation, we provide a basis for explaining how machine learning systems use information. These explanations increase trust in the functioning of the system, allowing us to verify that they make not only right decisions but also for justifiable reasons. Further, explanations can be used to support detection of privacy and fairness violations, as well as explain how they came about. We can then leverage this understanding to repair systems to avoid future violations.

We identify two major challenges to explaining information use in machine learning systems: (i) *converged use*, that machine learning systems typically combine a large number of input features, and (ii) *indirect use*, that these systems can typically infer and use information that is not directly provided to the system. Our approach to explaining how complex machine learning models use information involves answering two questions: (influence) Which factors were influential in determining outcomes?, and (interpretation) What do these factors mean? We first present key results measuring the causal influence of factors in machine learning models. We then examine the following settings: (i) systems with potential indirect use of information, and (ii) convolutional neural networks. For each setting we demonstrate how influence and interpretation combine to account for information use.

# Acknowledgements

As a person who is generally susceptible to external influences it was quite essential that the influences that shaped grad school for me were positive. They were overwhelmingly so, and as a result, I have many people to thank. The sum of their teaching, contributions, advice and encouragement is what propelled me through six years of graduate studies.

Anupam Datta taught me that the discovery of science is hardly ever linear. It is a series of non-linear jumps that when strung together lead to the discovery of something beautiful. He also taught me how to write better, present clearly, teach with passion and think deeper. Anupam's guidance and friendship have left a deep imprint on who I am and who I'll be.

I'm also fortunate to have been guided by an illustrious committee comprising of Jaime Carbonell, Matt Fredrikson, Sriram Rajamani and Jeannette Wing. Their advice has greatly improved both the ideas in this thesis and its writing. Even before they were part of my committee, my work has been deeply influenced by my collaboration and interactions with them. A phone call from Jeannette in the middle of an undergrad networks class in 2011 is what started my CMU journey in the first place. Sriram's energetic mentorship across two internships at Microsoft Research India has shaped the directions my research has taken. Every meeting with Matt is a fountain of great ideas. Jaime's deep insights have refined the ideas in this thesis.

I've been lucky to have worked with some amazing collaborators who approximately in order of appearance are: Joe Halpern, Saikat Guha, Sriram Rajamani, Deepak Garg, Limin Jia, Aditya Nori, Aleksandar Chakarov, Yair Zick, Michael Tschantz, Matt Fredrikson, Piotr Mardziel, Gihyuk Ko, Klas Leino, Sophia Kovaleva, and Linyi Li. A special shout out to Saikat: my internship with him is what led me down the trail of ideas that eventually turned into this thesis.

When I started off teaching, I used to be terrible. Now I'm slightly less terrible. Bob Harper, Andre Platzer, Anupam Datta, and Sid Jain are to thank.

CMU has a wonderful support staff that keep things running under the hood that make the place work like magic. In particular, I'd like to thank Deb Cavlovich, Martha Clarke, Catherine Copetas, Marcella Baker, Diana Leathers and Kelley Conley.

A wonderful group of friends have kept me sane through the years through crosswords, beers, road trips and food. These are (in no particular order) Rajesh Bhattacharjee, Alejandro Carbonara, Shriphani Palakodety, Ashique Khudabukhsh, Ayantika Ghosh, Amit Datta, Sid Jain, Sahil Singla, Dougal Sutherland, David Kurokawa, Anindit Mukherjee, Kristina Sojakova, Sam Yeom, Sohinee Bhatacharyya and Aram Ebtekar.

Any description of the impact Sumedha Roy has had on this thesis will be an understatement. Over the years she has been a proofreader, cheerleader, life coach, thesaurus, travel companion, the love of my life and now, my wife.

My parents have been a infinite source of support, guidance and love. This thesis is dedicated to them.

# Contents

# Chapter 1

# Introduction

Machine learning systems are increasingly being deployed in application domains that significantly affect people's lives. Examples of such domains include credit, insurance, healthcare, personalized recommendations, and criminal justice [98, 141, 143]. Yet complex machine learning models that are being pushed into production remain opaque—it is difficult to explain to human users why a model made certain predictions. Indeed automated loan denials [141], radiology image classification [143] and prison sentence recommendations [98] are usually not accompanied by any explanation of how such decisions are made. This situation is problematic because models may not be making predictions for justifiable reasons. Their predictions over the long run may thus be untrustworthy (i.e., exhibit poor accuracy), and pose a threat to societal values like privacy and fairness. This problem has been widely recognized over the past few years leading to calls to enhance accountability of machine learning systems [8, 13, 32, 47].

Machine learning models are optimized to make accurate predictions that best fit available data, without constraining the reasons behind the predictions they make. For example, in [111], Ribeiro et al., highlight an instance of the problem where a neural network identifies an image of a husky correctly, but only because of the snow that surrounds it. This neural network, however, will not be able to identify huskies outside of their snowy environment. A credit classifier might find zip-code to be predictive of credit-worthiness because of past biases encoded in datasets. However, the use of zip-code admits the potential for the indirect use of race as the two are often correlated, and lead to discriminatory outcomes. Similarly, an advertising system might discover that searching for certain health conditions is useful for displaying ads for therapies. A personalized advertising system that uses health information to target ads is concerning and considered to be an illegal privacy violation in certain jurisdictions [104]. Privacy and discrimination harms in particular have led to a requirement for explanations being mandated by a number of laws and regulations such as the Equal Credit Opportunity Act [4], the Fair Information Practice Principles (FIPPs) [131] in the USA, and the the General Data Protection Regulation in the EU [46].

All of the concerns above can be viewed as instances of improper *information use*: the information used to make decisions was either unreliable, or disallowed by privacy and fairness norms. In this dissertation, we provide a basis for developing explanations for information use in machine learning systems. Human experts can observe these explanations

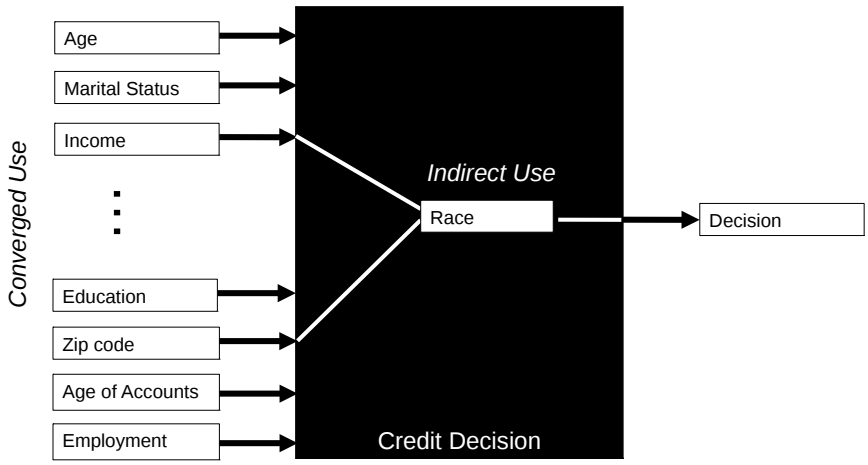



Figure 1.1: A schematic representation of a typical machine learning model illustrating the converged use problem that models typically have a large number of inputs with some influence, and the indirect use problem that models can combine information to infer attributes not directly provided as input.

in order to increase trust in the functioning of the system, allowing one to verify that they make not only right decisions but also for justifiable reasons. Further, explanations can be used to surface the use of protected information types such as race or health status in support of the detection of privacy and fairness violations. We can then leverage this understanding to repair systems to avoid future violations.

> Explanations of information use are a means to enhancing trust in the predictions of machine learning models, and surfacing violations of privacy and fairness.

A simple approach to explaining how models use information is to examine which inputs are used by a model. For example, if we were to examine if a credit decisioning system uses an input corresponding to age, we would vary that input while keeping all other inputs fixed and see if the outcome changes. Such a notion of information use, will be familiar to some readers as interference or causal influence.

However, machine learning models are typically a complex combination of a large number of inputs. Figure 1.1 illustrates that the simple approach of identifying which inputs are used by the model faces two key challenges. The first challenge is *converged use*, where each input (out of a large number) has some influence on the outcomes, and therefore a qualitative notion of influence suggests an overwhelmingly large number of factors. In the credit example above, which may combine information from credit reports with other account information, each factor is likely to have some, influence on the outcome. The second challenge is *indirect use*, where multiple inputs may be combined to infer information that wasn't directly provided to the model as input. For example, attributes protected against discrimination such as race could be inferred from a person's zip code, and used. Race wasn't provided as input, but was inferred and used indirectly.

We address the converged use problem by proposing a quantitative measure of influence

that computes the relative importance of different input factors in a model. Using a quantitative notion of influence allows examining the most important factors used by the model. In the example, in Figure 1.1, for a particular individual's decision income and employment might be the most important factors, while other factors might be of minor importance.

We address the indirect use problem by examining internal computations within a model and interpreting them to identify what information has been inferred by an internal computation. Further, as there may be many internal computations, we use quantitative influence measures developed to address the converged use problem to examine the most important influential factors, and then identifying the information inferred by them.

We examine this problem of indirect use in two settings. The first setting is motivated by privacy and non-discrimination requirements that prohibit the use of protected information types. Given some protected information type, we identify internal computations that are strongly associated with that information type. The second setting, is motivated by increasing trust, where we wish to discover the concepts being inferred by a model to make decisions. In this setting, we focus on convolutional neural networks, which are heavily used in computer vision tasks. For such models, input factors correspond to pixels in images and not to high level concepts, and therefore require us to examine concepts inferred by internal computations.

We term this general approach of explaining how information is used by a system by examining influential internal or external factors *influence-directed explanation*. The key thesis behind the approach can be stated as follows.

> Interpreting influential causal factors provides a principled basis for tools that explain how information is used in machine learning systems.

The theoretical and definitional results of this thesis are supported by algorithms implementing these results. In particular, the algorithms addressing the converged use problem scale to state-of-the-art models such as deep convolutional neural networks and large random forests. Scalability is still a concern for the algorithms addressing proxy use, a challenge that we plan to address in future work.

We now describe the main results in support of this thesis.

## 1.1 Quantifying Direct Use

While explaining a certain behavior of a system, identifying factors with causal influence on the behavior allows us to focus on the part of the system that is relevant. A qualitative notion of causal influence is identical to the notion of interference: changing an input while keeping all other inputs fixed, changes the output. Verifying the absence of interference has been the subject of much research, starting from the work of Denning [38]. However, for complex machine learning models that combine a large number of inputs, a qualitative notion of influence is too coarse grained as almost all inputs have some degree of influence, a problem that we term converged use. Therefore, we focus on quantifying the relative degree of influence of different input factors.

| | |
|---|---|
| Age | 23 |
| Workclass | Private |
| Education | 11th |
| Education-Num | 7 |
| Marital Status | Never-married |
| Occupation | Craft-repair |
| Relationship | Own-child |
| Race | Asian-Pac-Islander |
| Gender | Male |
| Capital Gain | 14344 |
| Capital Loss | 0 |
| Hours per week | 40 |
| Country | Vietnam |

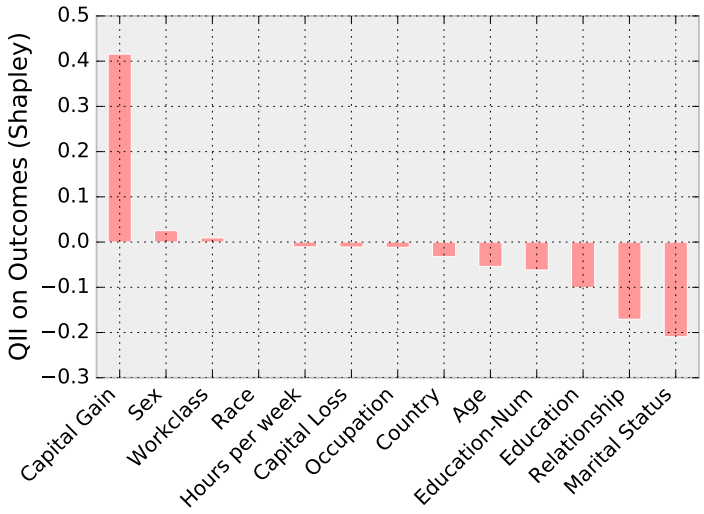



Figure 1.2: Using QII to explain negative classification for an individual.

In Chapter 2, we develop Quantitative Input Influence (QII), a family of measures to quantify the influence of inputs of systems on their outcomes. Distinctively, our causal QII measures carefully account for correlated inputs while measuring influence. For example, in 1.1, age and income may be correlated. As a result, even if a model makes decisions only based on income is bound to be correlated. QII measures are defined via causal interventions that break the correlations between inputs in order to expose the causal relationships between inputs and outputs.

Further, they support a general class of queries by being parametric in a quantity of interest about the system. By an appropriate parametrization of the quantity of interest, one can explain decisions about individuals (e.g., a loan decision) and groups (e.g., disparate impact based on gender).

Finally, single inputs may not always have high influence. For example, just changing an individual's income may not flip the decision. To address this problem, QII measures also quantify the joint influence of a set of inputs (e.g., age and income) on outcomes (e.g. loan decisions) and the marginal influence of individual inputs within such a set (e.g., income). Since a single input may be part of multiple influential sets, the average marginal influence of the input is computed using principled aggregation measures, such as the Shapley value, previously applied to measure influence in voting.

Figure 1.2 illustrates the use of QII to explain an individual's negative classification by a model, and exposes the key positive and negative factors behind it. In this example, while this individual's capital gains income was a strong positive factor, education, and marital status featured among the key negative factors that led to a negative classification. A particular use case we envision for this mode of explanation is to support adverse action notification requirements for regulations such as the Equal Credit Opportunity Act (ECOA) which requires institutions to provide the principal reasons behind any decision that adversely affects an individual.

Under most circumstances, QII measures can be approximated efficiently: for a fixed degree of accuracy, QII can be computed in linear time in the number of features for bounded quantities of interest. We also demonstrate that even for complex random forest models, influence is concentrated in a small number of features for any individual, thus satisfying requirements of conciseness in many settings such as adverse action notices in credit [4].

**Supervising Causal Influence.** In order to measure the causal influence of inputs, we violate *distributional faithfulness*, that is, we observe the outcome of the classifier on counterfactual points: points which change one input while keeping all other inputs fixed. Such causal experimentation is the staple of much of the natural sciences. However, in a problem peculiar to machine learning models, since counterfactual inputs may lie outside of distribution the model was trained on, the model is not required to behave meaningfully on such inputs. As a result, traditional machine learning methods do not constrain the causal influences of features. In Chapter 3, we formalize the conditions under which causal influences are unconstrained, and develop an active learning algorithm that constrains causal influences by training on counterfactual points, We term this approach *counterfactual active learning*.

**Influence for Differentiable Models.** For differentiable models, it is possible to measure the effect of infinitesimal interventions; computing partial derivatives involve changing an input with an infinitesimal amount while keeping all other features fixed. This infinitesimal intervention gives us distributional faithfulness for free if the points we are measuring the partial derivatives on lies within the distribution. In Chapter 4, we present *distributional input influence*, an axiomatically justified influence measure that leverages this observation to measure causal influences while retaining distributional faithfulness. This input influence measure is generalized in Chapter 5 to internal factors in models, and forms the basis for our explanations of the predictions of convolutional neural networks.

## 1.2 Quantifying Indirect Use

While the first part of this dissertation focuses on the converged use problem, a second challenge is that machine learning models can combine inputs to infer information not directly provided as input. In such settings only examining input factors is insufficient in explaining how these models use information. We explore the indirect use problem in two settings.

**Low-level input features.** In settings such as image classification, individual features (pixels) do not represent meaningful concepts. For models like deep networks, the system processes the low-level inputs into higher level intermediate features. In order to build trust in the predictions of the model, we surface influential intermediate features in the model, allowing a domain expert to judge whether the reasons behind a prediction are justifiable.

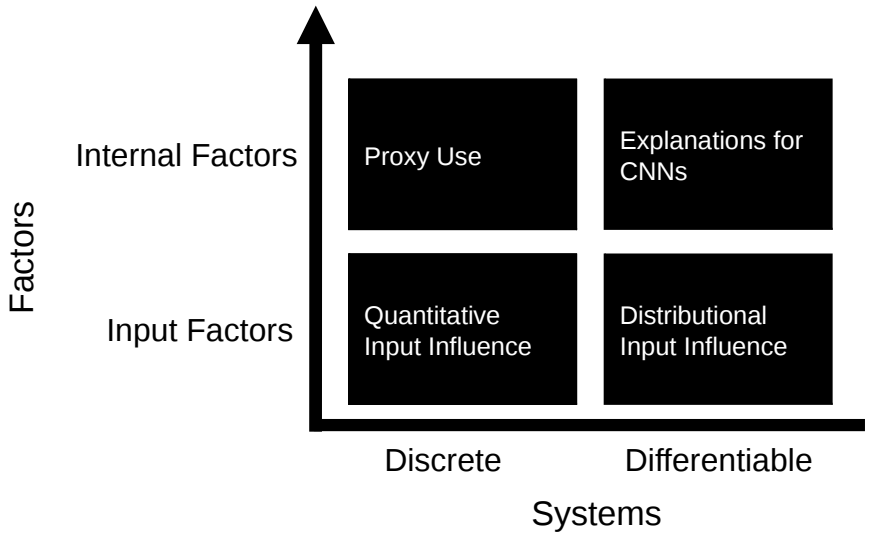


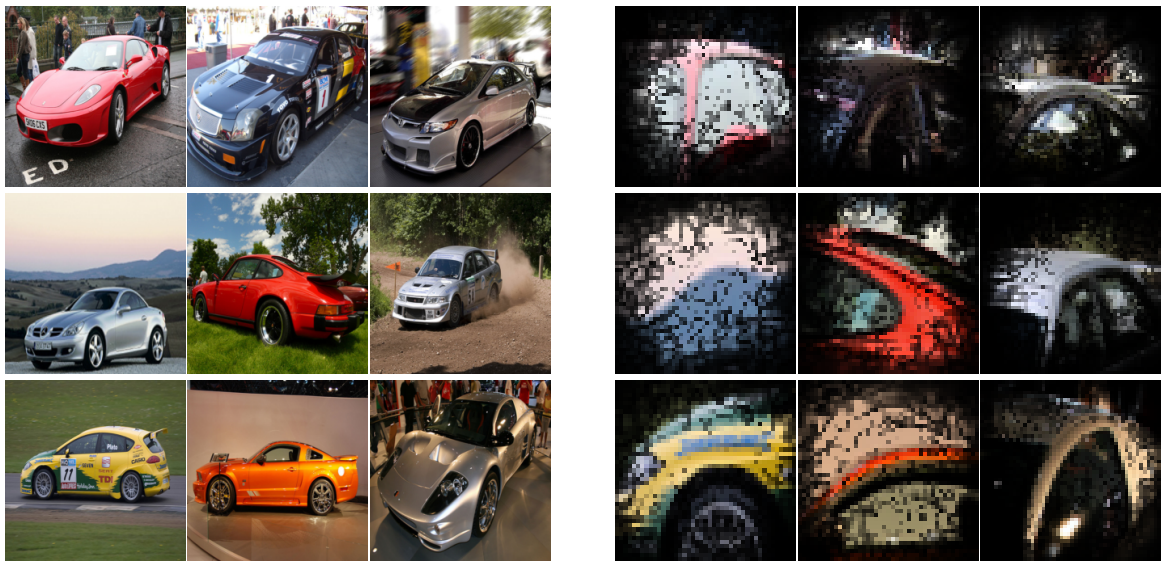

Figure 1.3: A summary of influence-directed explanations in different settings. For a setting where a discrete model such as a classifier has interpretable inputs, QII (Chapter 2) provides an explanation for system behaviors in terms of input influence. Our method for detecting proxy use (Chapter 6) identifies internal computations that are strongly associated with some protected attribute and have causal influence on the outcome. For neural networks (Chapter 5), influence directed explanations identify influential neurons that can be interpreted with appropriate visualization techniques.

**Proxies.** Information may be used indirectly through proxies even if it isn't directly provided to the system, and violate privacy and non-discrimination norms that prevent the use of protected information types. Identifying information use through proxies, and eliminating them entails accounting for associations that may be present between features and sensitive attributes.

In both of these scenarios, we propose explanation tools that identify intermediate factors with causal influence on outcomes. Since a model might have a large number of intermediate factors, we use quantitative influence measures to focus on the most important ones. Further, these intermediate factors are themselves complex computations and it may not readily apparent what information they correspond to. In Figure 1.1, the intermediate computation is not apriori labeled with the information type 'race'. As a result, we provide a mechanism for associating the influential intermediate factors with human-understandable concepts. Figure 1.3 summarizes the main approaches to influence-directed explanations.

### 1.2.1 Explanations for CNNs

In settings such as image classification with deep neural networks, individual input factors (such as pixels) do not represent meaningful concepts. Deep neural networks, therefore,



**(a)** **(b)**

Figure 1.4: Images of cars labeled "sports car" by VGG16 ImageNet model (a) and receptive fields of the most influential feature map on a comparative quantity that characterizes the model's tendency to predict "sports car" over "convertible" (b). In most cases, the features within the field contain the top of the car, which is the key distinctive concept between these classes.

process low-level input features into representations organized in layers that get progressively richer with the depth of layers, and then use these representations to make predictions. However, due to the complexity of these models, which typically have millions of parameters, it is difficult to build trust that these models make decisions for justifiable reasons, and debug reasons when they make mistakes.

In Chapter 5, we study the problem of explaining a rich class of behavioral properties of deep neural networks. In line with the general template in this thesis, our *influence-directed explanations* approach this problem by peering inside the network to identify neurons with high *influence* and then providing an *interpretation* for the concepts these neurons represent.

As an example, Figure 1.4 demonstrates on a CNN model [121] trained on the ImageNet dataset [114] the capability of influence-directed explanations to extract meaningful insight about the network's inner workings. We measure the influence of neurons at an internal layer on the network's tendency to predict "sports car" over "convertible". The images in Figure 1.4(b) are computed by visualizing the most influential neuron for the corresponding image in 1.4(a). The results coincide with an intuitive understanding of the distinction between these classes: the depicted interpretation highlights the portion of the image depicting the car's top.

We introduce a novel distributional influence measure that allows us to identify which neurons are most influential in determining the model's behavior on a given distribution

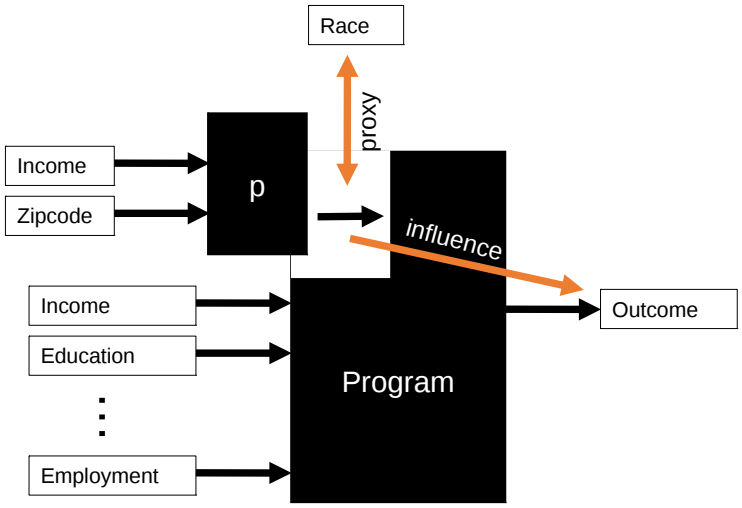



Figure 1.5: A visual representation of the proxy use definition that requires identifying an internal factor that is strongly associated with a protected attribute, and has influence on the outcome.

of instances. From this we are able to identify the learned concepts that cause the network to behave characteristically, for example, on the distribution of instances that share a particular label. This distributional influence measures shares key attributes with QII. In particular, the measure accounts for causal influences and is parametric in a quantity of interest. However, since neural networks are usually continuous and differentiable models, it is possible to examine the effects of infinitesimally changing a factor. In effect, these infinitesimal interventions allows us to measure causal effects without evaluating the model on points outside the distribution containing input instances.

These influence-directed explanations can be approximated with a constant number of gradient calls, where the constant depends on the number of parameters, and the accuracy required. Gradient calls are relatively inexpensive even for large networks through optimized parallel implementations on GPUs, allowing us to scale to state-of-the-art convolutional neural networks.

### 1.2.2 Proxy Use

Information can be used indirectly through proxies, which allow a data processor to effectively infer protected information types and use them even when they are not explicitly provided. In Figure 1.1, we discussed an example where income and zip code could be combined to infer race. Such indirect uses run afoul of privacy and non-discrimination norms that govern the use of protected information types. For example, the Canadian Personal Information Protection and Electronic Documents Act (PIPEDA) [139], prohibits the use of health information for advertising and marketing. Similarly, for non-discrimination, the Equal Credit Opportunity Act (ECOA) [4] prohibits the use of information types such as race, or gender in lending decisions.

In Chapter 6, we propose a theory of proxy use, and use it as a building block to construct theories of *use privacy* and *proxy non-discrimination*. The definition of proxy use, illustrated in Figure 1.5, identifies intermediate factors in a program that has (i) causal influence on outcomes (influence), and (ii) is strongly associated with a protected attribute (interpretation). We arrive at this program-based definition after a careful examination of the space of possible definitions. In particular, we prove that it is impossible for a purely semantic notion of intermediate computations to support a meaningful notion of proxy use as characterized by a set of natural properties or axioms.

Along with the definition, we provide algorithms for the detection and repair of proxy use for a number of standard machine learning models such as logistic regression, decision trees, and tree ensembles. We evaluate the performance of the detection algorithm and show that, in particular cases, the runtime of our system scales linearly in the number of intermediate computations in a model. For models with a large number of intermediate computations such as linear models, the algorithms do not scale beyond small models. Improving the scalability of our proxy use detection and repair algorithms is an important direction of future work. In a recent unpublished manuscript [76], we report a novel model checking approach to the problem that increases the size of models that can be verified by up to 5x and reduces verification time by up to 13x over a baseline algorithm for this task.

The focus on use is a significant departure from a large body of prior work that focuses on limiting disclosures for privacy (see [124] for a survey), and disparate impact for fairness [50, 70, 105, 135, 149], which can both be viewed forms of probabilistic association. We now describe how the definition of proxy use forms a building block for new, meaningful definitions of privacy and non-discrimination.

**Use Privacy** Use privacy constraints restrict the use of protected information types and some of their proxies.

A use privacy constraint may require that health information or its proxies not be used for advertising. Indeed, there are calls for this form of privacy constraint [37, 87, 100, 139]. In this work, we consider the setting where a system is audited to ensure that it complies with such use privacy constraints. The auditing could be done by a co-operative data processor who is operating the system or by a regulatory oversight organization who has access to the data processors' machine learning models and knowledge of the distribution of the dataset. In other words, we assume that the data processor does not act to evade the detection algorithm, and provides accurate information.

In this setting, it is impossible to guarantee that data processors with strong background knowledge are not able to infer certain facts about individuals [43]. Even in practice, data processors often have access to detailed profiles of individuals and can infer sensitive information about them [41, 138]. Use privacy instead places a more pragmatic requirement on systems: that they simulate ignorance of protected information types by not using them or their proxies in their decision-making. This requirement is met if the systems (e.g., machine learning models) do not infer protected information types or their proxies (even if they could) or if such inferences do not affect decisions.

Recognizing that not all instances of proxy use of a protected information type are

inappropriate, our theory of use privacy makes use of a normative judgment oracle that makes this inappropriateness determination for a given instance. For example, while using health information or its proxies for credit decisions may be deemed inappropriate, an exception could be made for proxies that are directly relevant to the credit-worthiness of the individual (e.g., her income and expenses).

**Proxy Non-discrimination** Analogous to use privacy, proxy non-discrimination constraints restrict the use of protected information types such as gender, race and nationality for purposes such as credit, insurance and healthcare.

Two popular approaches to addressing the problem of discrimination are the prevention of disparate impact and disparate treatment. Disparate impact identifies cases where group parity is violated i.e., where the fraction of individuals who get positive outcomes are very different across protected and unprotected groups in the population. The 80% rule in hiring and promotions is an embodiment of this approach that can be traced back to the Griggs v. Duke Power ruling [20]. However, it has been pointed out that the group parity often does not ensure outcomes which are fair [45]. On the other hand disparate treatment, rules out explicit uses of protected information, which does not rule out inferences of protected information being used. Instead, as with privacy, we take a pragmatic approach of detecting evidence of proxy use of protected information.

## 1.3 Related Work

Existing tools that explain information use employ either causal techniques (interference, information use, causal testing, differential privacy) or associative techniques (quantitative information flow, information leakage, disparate impact testing). Instead of choosing a purely causal or associative view, this dissertation proposes an novel approach that combines the two. We use causal techniques to identify influential factors, and associative ones to interpret them. We categorize some closely related work here by area: information flow security, privacy and fairness.

**Information Flow Security.** There has been significant research activity in restricting information flows in programs over the last three decades, and language-based methods to support these restrictions ([38, 95, 107]). These methods enforce non-interference or variants of it from sensitive inputs of the program to outputs.

Work on quantifying information flow has largely focused on quantifying the leakage of information about sensitive attributes to an adversary. Quantitative Information Flow is concerned with information leaks and therefore needs to account for correlations between inputs that may lead to leakage, making measures of associations between inputs and outcomes appropriate (see [123] for an overview). On the other hand, we take the position that information use is a causal notion, and therefore measuring it requires destroying correlations through interventions.

Finally, a line of work on information use and information flow experiments [32, 33, 136], formalizes the relation between causality, probabilistic non-interference, and information

use, and develops a framework for black-box experimentation on web services. Black-box experimentation is an important approach to achieving accountability in data driven systems.

**Privacy in Statistical Systems.** Privacy in the presence of data analytics has largely focused on minimizing the disclosure of personal information. Differential privacy [44] and its variants belong to this class of properties in a setting with a trusted data processor and an untrusted adversary trying to infer sensitive information about individuals. Differential privacy provides the guarantee that any adversary will gain approximately the same information with or without an individual's participation in a dataset. Other formal properties related to privacy focus on limiting the flow of information using notions such as statistical disclosure limitation [48], characterizing possible inferences from data releases [30, 40, 115], or that your participation in a study should not become known [62].

Our notion of use privacy is quite complementary to this body of prior work. Instead of trying to limit disclosures through system outputs, we focus instead on ensuring that protected information types and their proxies are not used internally by the data analytics system, and could be used in conjunction with methods that limit disclosures of sensitive information.

**Fairness in Statistical Systems.** Recently, the algorithmic foundations of fairness in personal information processing systems have received significant attention [22, 32, 45, 70, 149]. While many of the algorithmic approaches [22, 70, 149] have focused on group parity as a metric for achieving fairness in classification, Dwork et al. [45] argue that group parity is insufficient as a basis for fairness, and propose a similarity-based approach which prescribes that similar individuals should receive similar classification outcomes, along with algorithms for achieving this by design. However, this approach requires a similarity metric for individuals which is often subjective and difficult to construct.

Instead of trying to achieve fairness by design, in our theory of proxy non-discrimination, we attempt to detect and remove instances of discrimination arising out of identifiable explicit or proxy use of protected attributes.

In this thesis, we will focus on privacy and fairness harms that arise out of improper information use. Limitations on information use are already well recognized norms in the domains described above. For example, the *use limitation* norms in law and guidelines such as the FTC's FIPPs in United States [131], the PIPEDA in Canada [104], and the GDPR in the European Union [46], require information use to only be limited to the purposes for which it was collected, and additionally restrict the use of sensitive information types such as health status, and sexual orientation. Anti-discrimination laws in employment [3], housing, credit [4] prevent the use of protected attributes such as gender, race, nationality, and sexual orientation for making decisions.

# Part I

# Converged Use

# Chapter 2

# Quantifying Input Influence

Machine learning models are typically a complex combination of a large number of inputs. In order to understand what information is being used by a model, the first challenge, which we term *converged use*, is that each input (out of a large number) has some influence on the outcomes. As a result, a qualitative notion of influence suggests an overwhelmingly large number of factors.

We introduce a family of *Quantitative Input Influence (QII)* measures that capture the degree of relative influence of inputs on outputs of the system. Three desiderata drive the definitions of these measures.

First, we seek to answer *general* class of related to input influence, ranging from specific questions about individuals to general questions about groups.

Second, we seek input influence measures that appropriately account for *correlated inputs*—a common case for our target applications. For example, consider a system that assists in hiring decisions for a moving company. Gender and the ability to lift heavy weights are inputs to the system. They are positively correlated with one another and with the hiring decisions. Yet whether the system uses the weight lifting ability or the gender in making its decisions (and to what degree) has substantive implications for determining if it is engaging in discrimination (the business necessity defense could apply in the former case [1]). This observation makes us look beyond correlation coefficients and other associative measures.

Third, we seek measures that appropriately quantify input influence in settings where any input by itself does not have significant influence on outcomes but a set of inputs does. In such cases, we seek measures of the *joint influence* of a set of inputs (e.g., age and income) on a system's decision (e.g., to serve a high-paying job ad). We also seek measures of the *marginal influence* of an input within such a set (e.g., age) on the decision. This notion allows us to measure the relative importance of individual inputs within the set (e.g., age vs. income) in the system's decision.

We achieve the first desideratum by formalizing a notion of a *quantity of interest*. The influence of an input is parameterized by a quantity of interest, which represents a property of the system for a given input distribution. Our formalization supports a wide range of statistical properties including probabilities of various outcomes in the output distribution and probabilities of output distribution outcomes conditioned on input distribution events.

Examples of quantities of interest include the conditional probability of an outcome for a particular individual or group, and the ratio of conditional probabilities for an outcome for two different groups (a metric used as evidence of disparate impact under discrimination law in the US [1]).

We achieve the second desideratum by formalizing *causal* QII measures. These measures (called *Unary QII*) model the difference in the quantity of interest when the system operates over two related input distributions—the real distribution and a hypothetical (or counterfactual) distribution that is constructed from the real distribution in a specific way to account for correlations among inputs. Specifically, if we are interested in measuring the influence of an input on a quantity of interest of the system behavior, we construct the hypothetical distribution by retaining the marginal distribution over all other inputs and sampling the input of interest from its prior distribution. This choice breaks the correlations between this input and all other inputs and thus lets us measure the influence of this input on the quantity of interest, independently of other correlated inputs. Revisiting our moving company hiring example, if the system makes decisions only using the weightlifting ability of applicants, the influence of gender will be zero on the ratio of conditional probabilities of being hired for males and females.

We achieve the third desideratum in two steps. First, we define a notion of joint influence of a set of inputs (called *Set QII*) via a natural generalization of the definition of the hypothetical distribution in the Unary QII definition. Second, we define a family of *Marginal QII* measures that model the difference on the quantity of interest as we consider sets with and without the specific input whose marginal influence we want to measure. Depending on the application, we may pick these sets in different ways, thus motivating several different measures. For example, we could fix a set of inputs and ask about the marginal influence of any given input in that set on the quantity of interest. Alternatively, we may be interested in the average marginal influence of an input when it belongs to one of several different sets that significantly affect the quantity of interest. We consider several marginal influence aggregation measures from cooperative game theory originally developed in the context of influence measurement in voting scenarios and discuss their applicability in our setting. We also build on that literature to present an efficient approximate algorithm for computing these measures.

Recognizing that different forms of influence may be appropriate for different settings, we generalize our QII measures to be parametric in its key elements: the intervention used to construct the hypothetical input distribution; the quantity of interest; and the difference measure used to quantify the distance in the quantity of interest when the system operates over the real and hypothetical input distributions. This generalized definition provides a structure for exploring the design space of influence measures.

Since transparency reports released to an individual, regulatory agency, or the public might compromise individual privacy, we explore the possibility of answering transparency queries while protecting differential privacy. We prove bounds on the sensitivity of a number of transparency queries and leverage prior results on privacy amplification via sampling [71] to accurately answer these queries.

We illustrate the utility of the QII framework by developing three machine learning applications on real datasets: an income classification application based on the benchmark

`adult` dataset [84], a predictive policing application based on the National Longitudinal Survey of Youth [2], and a credit application based on the Lending Club dataset from Kaggle [81]. Using these applications, we argue, in Section 2.6, the need for causal measurement by empirically demonstrating that in the presence of correlated inputs, observational measures are not informative in identifying input influence. In particular, we illustrate for these applications how QII can provide insights into individual and group classification outcomes. We also demonstrate that under most circumstances, QII measures can be made differentially private with minimal addition of noise, and can be approximated efficiently: for a fixed degree of accuracy, QII can be computed in linear time in the number of features for bounded quantities of interest. Finally, we demonstrate that for even for complex random forest models, influence is concentrated in a small number of features for any individual, thus satisfying requirements of conciseness in many settings such as adverse action notices in credit [4].

## 2.1 Unary QII

Consider the situation discussed in the introduction, where an automated system assists in hiring decisions for a moving company. The input features used by this classification system are : *Age*, *Gender*, *Weight Lifting Ability*, *Marital Status* and *Education*. Suppose that, as before, weight lifting ability is strongly correlated with gender (with men having better overall lifting ability than women). One particular question that an analyst may want to ask is: "What is the influence of the input *Gender* on positive classification for women?". The analyst observes that 20% of women are approved according to his classifier. Then, she replaces every woman's field for gender with a random value, and notices that the number of women approved does not change. In other words, an *intervention* on the *Gender* variable does not cause a significant change in the classification outcome. Repeating this process with *Weight Lifting Ability* results in a 20% increase in women's hiring. Therefore, she concludes that for this classifier, *Weight Lifting Ability* has more influence on positive classification for women than *Gender*.

By breaking correlations between gender and weight lifting ability, we are able to establish a causal relationship between the outcome of the classifier and the inputs. We are able to identify that despite the strong correlation between a negative classification outcome for women, the feature *Gender* was not a cause of this outcome.

A mechanism for breaking correlations to identify causal effects is called an intervention in the literature on causality. In this chapter, we introduce a particular randomized intervention, and we describe in Section 2.2 how other interventions may be useful depending on the causal question asked of the system.

In the example above, instead of the entire population of women, the analyst may be interested in a particular individual, say Alice, and ask "What is the influence of the input *Gender* on Alice's rejection?". The analyst answers this question by applying the same intervention, but only on Alice's data, and observes that keeping every other input feature fixed and replacing *Gender* with a random value has no effect on the outcome.

In general, QII supports reasoning at such different scales by being parametric in a

*quantity of interest*. A quantity of interest is a statistic of the system, which represents the subject of the causal question being asked. For example, a suitable quantity of interest corresponding to the question "What is the influence of the input *Gender* on positive classification for women?" is average rate of positive outcomes for women. Similarly, a suitable quantity of interest for "What is the influence of the input *Gender* on Alice's rejection?" is just Alice's classification outcome. QII also supports reasoning about more complex statistics such ones which measure group disparity. For example, the analyst observes that the rate of positive classification is 20% higher in men than in women, and wishes to ask "What is the influence of *Income* on men getting more positive classification than women?". In this case, a natural quantity of interest is the difference in the classification rates of men and women.

We now present the formal definition of QII.

We are given an algorithm $\mathcal{A}$. $\mathcal{A}$ operates on inputs (also referred to as *features* for ML systems), $N = \{1, \ldots, n\}$. Every $i \in N$, can take on various *states*, given by $X_i$. We let $\mathcal{X} = \prod_{i \in N} \mathcal{X}_i$ be the set of possible feature state vectors, let $\mathcal{Z}$ be the set of possible outputs of $\mathcal{A}$. For a vector $\mathbf{x} \in \mathcal{X}$ and set of inputs $S \subseteq N$, $\mathbf{x}|_S$ denotes the vector of inputs in $S$. Inputs are assumed to be drawn from some probability distribution that represents the population. The random variable $X$ represents a feature state vector drawn from this probability distribution.

For a random variable $X$ representing the feature vector, a *randomized intervention* on feature $i$ is denoted by the random variable $X_{-i}U_i$, where $U_i$ is new random variable that is independently drawn from the prior distribution of feature $i$. For example, to randomly intervene on age, we replace the age for every individual with a random age from the population.

A *quantity of interest* $Q_{\mathcal{A}}(X)$ of some algorithm $\mathcal{A}$ is a statistic of the algorithm over the population $X$ is drawn from. As discussed above, examples of quantities of interest are the average rate of positive classification for a group, or the classification outcome of an individual.

The Quantitative Input Influence of an input on a quantity of interest is just the difference in the quantity of interest with or without the intervention.

**Definition 1** (QII)**.** *For a quantity of interest $Q_{\mathcal{A}}(\cdot)$, and an input $i$, the Quantitative Input Influence of $i$ on $Q_{\mathcal{A}}(\cdot)$ is defined to be*

$$\iota^{Q_{\mathcal{A}}}(i) = Q_{\mathcal{A}}(X) - Q_{\mathcal{A}}(X_{-i}U_i).$$

We now instantiate this definition with different quantities of interest to illustrate the above definition in three different scenarios.

**QII for Individual Outcomes** One intended use of QII is to provide personalized transparency reports to users of data analytics systems. For example, if a person is denied a job application due to feedback from a machine learning algorithm, an explanation of which factors were most influential for that person's classification can provide valuable insight into the classification outcome.

For QII to quantify the use of an input for individual outcomes, we define the quantity of interest to be the classification outcome for a particular individual. Given a particular

individual $\mathbf{x}$, we define $Q^{\mathbf{x}}_{\text{ind}}(\cdot)$ to be $\mathbb{E}(c(\cdot) = 1 \mid X = \mathbf{x})$. The influence measure is therefore:

$$\iota^{\mathbf{x}}_{\text{ind}}(i) = \mathbb{E}(c(X) = 1 \mid X = \mathbf{x}) - \mathbb{E}(c(X_{-i}U_i) = 1 \mid X = \mathbf{x}) \tag{2.1}$$

When the quantity of interest is not the probability of positive classification but the classification that $\mathbf{x}$ actually received, a slight modification of the above QII measure is more appropriate:

$$\begin{aligned} \iota^{\mathbf{x}}_{\text{ind-act}}(i) &= \mathbb{E}(c(X) = c(\mathbf{x}) \mid X = \mathbf{x}) - \mathbb{E}(c(X_{-i}U_i) = c(\mathbf{x}) \mid X = \mathbf{x}) \\ &= 1 - \mathbb{E}(c(X_{-i}U_i) = c(\mathbf{x}) \mid X = \mathbf{x}) = \mathbb{E}(c(X_{-i}U_i) \neq c(\mathbf{x}) \mid X = \mathbf{x}) \end{aligned} \tag{2.2}$$

The above probability can be interpreted as the probability that feature $i$ is pivotal to the classification of $c(\mathbf{x})$. Computing the average of this quantity over $X$ yields:

$$\textstyle\sum_{\mathbf{x} \in \mathcal{X}} \Pr(X = \mathbf{x})\, \mathbb{E}(i \text{ is pivotal for } c(X) \mid X = \mathbf{x}) = \mathbb{E}(i \text{ is pivotal for } c(X)). \tag{2.3}$$

We denote this average QII for individual outcomes as defined above, by $\iota_{\text{ind-avg}}(i)$, and use it as a measure for importance of an input towards classification outcomes.

**QII for Group Outcomes** For more general findings, the quantity of interest may be the classification outcome for a set of individuals. Given a group of individuals $\mathcal{Y} \subseteq \mathcal{X}$ , we define $Q^{\mathcal{Y}}_{\text{grp}}(\cdot)$ to be $\mathbb{E}(c(\cdot) = 1 \mid X \in \mathcal{Y})$. The influence measure is therefore:

$$\iota^{\mathcal{Y}}_{\text{grp}}(i) = \mathbb{E}(c(X) = 1 \mid X \in \mathcal{Y}) - \mathbb{E}(c(X_{-i}U_i) = 1 \mid X \in \mathcal{Y}) \tag{2.4}$$

**QII for Group Disparity** Instead of simply classification outcomes, an analyst may be interested in more nuanced properties of data analytics systems such as the presence of disparate impact. Disparate impact compares the rates of positive classification within protected groups defined by gender or race. In employment, its absence is often codified as the '80% rule' which states that the rate of selection within a protected demographic should be at least 80% of the rate of selection within the unprotected demographic. The quantity of interest in such a scenario is the ratio in positive classification outcomes for a protected group $\mathcal{Y}$ from the rest of the population $\mathcal{X} \setminus \mathcal{Y}$.

$$\frac{\mathbb{E}(c(X) = 1 \mid X \in \mathcal{Y})}{\mathbb{E}(c(X) = 1 \mid X \notin \mathcal{Y})}$$

However, the ratio of classification rates is unstable at low values of positive classification. Therefore, for the computations in this chapter we use the difference in classification rates as our measure of group disparity.

| Quantity of Interest | Intervention | Difference |
|---|---|---|
| Individual Outcome | Constant | Subtraction |
| Average Outcome | Prior | Absolute Difference |
| Distribution | | Earthmover Distance |
| Group Disparity | | Mutual Information |

Table 2.1: Examples of influence schema elements

$$Q^{\mathcal{Y}}_{\text{disp}}(\cdot) = |\mathbb{E}(c(\cdot) = 1 \mid X \in \mathcal{Y}) - \mathbb{E}(c(\cdot) = 1 \mid X \notin \mathcal{Y})| \tag{2.5}$$

The QII measure of an input group disparity, as a result is:

$$\iota^{\mathcal{Y}}_{\text{disp}}(i) = Q^{\mathcal{Y}}_{\text{disp}}(X) - Q^{\mathcal{Y}}_{\text{disp}}(X_{-i}U_i). \tag{2.6}$$

More generally, group disparity can be viewed as an association between classification outcomes and membership in a group. QII on a measure of such association (e.g., group disparity) identifies the variable that causes the association in the classifier. *Proxy variables* are variables that are associated with protected attributes. For addressing concerns of discrimination such as *digital redlining*, it is important to identify which proxy variables actually introduce group disparity. It is straightforward to observe that features with high QII for group disparity are proxy variables that cause group disparity. Therefore, QII on group disparity is a useful diagnostic tool for determining discrimination. The use of QII in identifying proxy variables is explored experimentally in Section 2.6.2.

## 2.2 Influence Schema

The forms of explanations discussed so far are specific instances of a general schema for input influence that we describe now. Recall that the influence of an input is the difference in a quantity of interest before and after an intervention. This definition features the following three elements of our influence schema.

- A *quantity of interest*, which captures the aspect of the system we wish to explain.
- An *intervention distribution*, which defines how the alternate distribution is constructed from the true distribution.
- A *difference measure*, which quantifies the difference between two quantities of interest.

The particular instantiation of elements in this schema is determined by the question the analyst wishes to answer about the system. Table 2.1 contains some examples of these instantiations and we describe the elements in detail below.

The first element, the *quantity of interest*, is determined by the the subject of the question. In the previous section, the systems considered were deterministic, and the quantities of interest considered were represented by a number such as an expectation or a probability. However, for more complex systems, the analyst may be interested in richer quantities about the system. For instance, if the system is randomized, a particular quantity of interest is the distribution over outcomes for a given individual.

The second element is the *causal intervention* used to compute influence. The particular randomized intervention considered in this chapter, $X_{-i}U_i$, where $U_i$ is drawn from the prior distribution of feature $i$, is suitable when no alternatives are suggested in the analyst's question. For example, the question "What is the influence of age on Bob's classification?" does not suggest any particular alternative age. On the other hand, when a particular alternative is suggested, a constant intervention is more suited. For example, for the question "Was age influential in Bob's rejection and Carl's acceptance?" Carl's age is the natural alternate to consider, and $U_i$ is drawn from the constant distribution corresponding to Carl's age.

Finally, we use a *difference measure* to compare the quantity of interest before and after the intervention. Apart from subtraction, considered above, examples of other suitable difference measures are absolute difference, and distance measures over distributions. The absolute difference measure can be employed when the direction of change of outcomes before and after an intervention is unimportant. When the quantity of interest is the distribution over outcomes, subtraction does not make sense, and a suitable measure of difference should be employed, such as the earthmover distance or the mutual information between the original and the intervened distributions.

## 2.3 Set and Marginal QII

In many situations, intervention on a single input variable has no influence on the outcome of a system. Consider, for example, a two-feature setting where features are age ($A$) and income ($I$), and the classifier is $c(A, I) = (A = \mathit{old}) \wedge (I = \mathit{high})$. In other words, the only datapoints that are labeled 1 are those of elderly persons with high income. Now, given a datapoint where $A = \mathit{young}, I = \mathit{low}$, an intervention on either age or income would result in the same classification. However, it would be misleading to say that neither age nor income have an influence over the outcome; changing both the states of income and age would result in a change in outcome.

Equating influence with the *individual* ability to affect the outcome is uninformative in real datasets as well: Figure 2.1 is a histogram of feature influence on individual outcomes for a classifier learned from the `adult` dataset [84][1]. For most individuals, all features have zero influence: changing the state of one feature alone is not likely to change the outcome of a classifier. Of the 19537 datapoints we evaluate, more than half have $\iota^{\mathbf{x}}(i) = 0$ for all $i \in N$; indeed, changes to outcome are more likely to occur if we intervene on *sets of features*. In order to better understand the influence of a feature $i \in N$, we should measure

[1]The `adult` dataset contains approximately 31k datapoints of users' personal attributes, and whether their income is more than \$50k per annum; see Section 2.6 for more details.

its effect when coupled with interventions on other features. We define the influence of a set of inputs as a straightforward extension of individual input influence. Essentially, we wish the influence of a set of inputs $S \subseteq N$ to be the same as when the set of inputs is considered to be a single input; when intervening on $S$, we draw the states of $i \in S$ based on the joint distribution of the states of features in $S$.

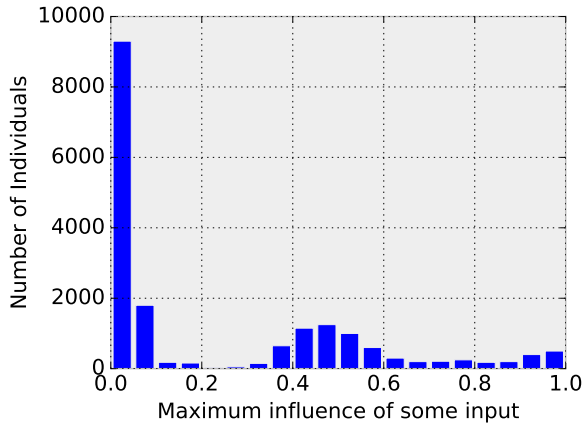



Figure 2.1: A histogram of the highest specific causal influence for some feature across individuals in the adult dataset. Alone, most inputs have very low influence.

We define the random variable $X_{-S}U_S$ representing an intervention on a set of features instead of a single feature. $X_{-S}U_S$ has the states of features in $N \setminus S$ fixed to their original values in $\mathbf{x}$, but features in $S$ take on new values according the joint marginal distribution over the features in $S$. This is identical to the intervention considered in unary QII, when the features in $S$ are viewed as a single composite feature. Using this generalization of an intervention to sets of features, we can define set QII that measures the influence of a set of features.

**Definition 2** (Set QII)**.** *For a quantity of interest $Q$, and an input $i$, the Quantitative Input Influence of set $S \subseteq N$ on $Q$ is defined to be*

$$\iota^Q(S) = Q(X) - Q(X_{-S}U_S).$$

Considering the influence of a set of inputs raises a number of interesting questions due to the interaction between inputs. First among these is how does one measure the *individual effect* of a feature, given the measured effects of interventions on sets of features. One natural way of doing so is by measuring the *marginal effect* of a feature on a set.

**Definition 3** (Marginal QII)**.** *For a quantity of interest $Q$, and an input $i$, the Quantitative Input Influence of input $i$ over a set $S \subseteq N$ on $Q$ is defined to be*

$$\iota^Q(i, S) = Q(X_{-S}U_S) - Q(X_{-S\cup\{i\}}U_{S\cup\{i\}}).$$

Notice that marginal QII can also be viewed as a difference in set QIIs: $\iota^Q(S \cup \{i\}) - \iota^Q(S)$. Informally, the difference between $\iota^Q(S\cup\{i\})$ and $\iota^Q(S)$ measures the "added value" obtained by intervening on $S \cup \{i\}$, versus intervening on $S$ alone.

The marginal contribution of $i$ may vary significantly based on $S$. Thus, we are interested in the *aggregate marginal contribution* of $i$ to $S$, where $S$ is sampled from some natural distribution over subsets of $N \setminus \{i\}$. In what follows, we describe a few measures for aggregating the marginal contribution of a feature $i$ to sets, based on different methods for sampling sets. The primary method of aggregating the marginal contribution is the Shapley value [120]. The less theoretically inclined reader can choose to proceed to Section 2.6 without a loss in continuity.

### 2.3.1 Cooperative Games and Causality

In this section, we discuss how measures from the theory of finite cooperative games define measures for aggregating marginal influence. In particular, we observe that the Shapley value [120] is uniquely characterized by axioms that are natural in our setting. Definition 2 measures the influence that an intervention on a set of features $S \subseteq N$ has on the outcome. One can naturally think of Set QII as a function $v : 2^N \to \mathbb{R}$, where $v(S)$ is the influence of $S$ on the outcome. With this intuition in mind, one can naturally study influence measures using *cooperative game theory*, and in particular, using the Shapley value. The Shapely value can be thought of as an *influence aggregation method*, which, given an influence measure $v : 2^N \to \mathbb{R}$, outputs a vector $\phi \in \mathbb{R}^n$, whose $i$-th coordinate corresponds in some natural way to the aggregate influence, or aggregate causal effect, of feature $i$.

The original motivation for game-theoretic measures is *revenue division* [91, Chapter 18]: the function $v$ describes the amount of money that each subset of players $S \subseteq N$ can generate; assuming that the set $N$ generates a total revenue of $v(N)$, how should $v(N)$ be divided amongst the players? A special case of revenue division that has received significant attention is the measurement of voting power [119]. In voting systems with multiple agents (think of political parties in a parliament), whose weights differ, voting power often does not directly correspond to agent weights. For example, the US presidential election can roughly be modeled as a finite cooperative game where each state is an agent. The weight of a state is the number of electors in that state (i.e., the number of votes it brings to the presidential candidate who wins that state). Although states like California and Texas have higher weight, swing states like Pennsylvania and Ohio tend to have higher power in determining the outcome of elections.

A voting system can be modeled as a finite cooperative game: players are voters, and the value of a coalition $S \subseteq N$ is 1 if $S$ can make a decision (e.g. pass a bill, form a government, or perform a task), and is 0 otherwise. Note the similarity to classification, with players being replaced by features (in classification, functions of the form $v : 2^N \to \{0, 1\}$ are referred to as boolean functions [99]). The game-theoretic measures of revenue division are a measure of *voting power*: how much influence does player $i$ have in the decision making process? Thus the notions of voting power and revenue division fit naturally with our goals when defining aggregate QII influence measures: in both settings, one is interested in measuring the aggregate effect that a single element has, given the actions of subsets.

Several canonical influence measures rely on the fundamental notion of *marginal contribution*. Given a player $i$ and a set $S \subseteq N \setminus \{i\}$, the marginal contribution of $i$ to $S$ is denoted $m_i(S, v) = v(S \cup \{i\}) - v(S)$ (we simply write $m_i(S)$ when $v$ is clear from the context). Marginal QII, as defined above, can be viewed as an instance of a measure of marginal contribution. Given a permutation $\pi \in \Pi(N)$ of the elements in $N$, we define $P_i(\sigma) = \{j \in N \mid \sigma(j) < \sigma(i)\}$; this is the set of $i$'s *predecessors* in $\sigma$. We can now similarly define the marginal contribution of $i$ to a permutation $\sigma \in \Pi(N)$ as $m_i(\sigma) = m_i(P_i(\sigma))$. Intuitively, one can think of the players sequentially entering a room, according to some ordering $\sigma$; the value $m_i(\sigma)$ is the marginal contribution that $i$ has to whoever is in the room when she enters it.

Generally speaking, game theoretic influence measures specify some reasonable way of

aggregating the marginal contributions of $i$ to sets $S \subseteq N$. In our setting, we argue for the use of the Shapley value. Introduced by the late Lloyd Shapley, the Shapley value is one of the most canonical methods of dividing revenue in finite cooperative games. It is defined as follows:

$$\varphi_i(N, v) = \mathop{\mathbb{E}}_{\sigma}[m_i(\sigma)] = \frac{1}{n!} \sum_{\sigma \in \Pi(N)} m_i(\sigma)$$

Intuitively, the Shapley value describes the following process: players are sequentially selected according to some randomly chosen order $\sigma$; each player receives a payment of $m_i(\sigma)$. The Shapley value is the expected payment to the players under this regime.

### 2.3.2 Axiomatic Treatment of the Shapley Value

In this work, the Shapley value is our function of choice for aggregating marginal feature influence. To justify our choice, we provide a brief exposition of axiomatic game-theoretic value theory. We present a set of axioms that uniquely define the Shapley value, and discuss why they are desirable in the QII setting.

The Shapley value satisfies the following properties:

**Definition 4** (Symmetry)**.** *We say that $i, j \in N$ are* symmetric *if $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subseteq N \setminus \{i, j\}$. A value $\phi$ satisfies* symmetry *if $\phi_i = \phi_j$ whenever $i$ and $j$ are symmetric.*

**Definition 5** (Dummy)**.** *We say that a player $i \in N$ is a* dummy *if $v(S \cup \{i\}) = v(S)$ for all $S \subseteq N$. A value $\phi$ satisfies the* dummy *property if $\phi_i = 0$ whenever $i$ is a dummy.*

**Definition 6** (Monotonicity)**.** *Given two games $\langle N, v_1\rangle, \langle N, v_2\rangle$, a value $\phi$ satisfies* strong monotonicity *if $m_i(S, v_1) \geq m_i(S, v_2)$ for all $S$ implies that $\phi_i(N, v_1) \geq \phi_i(N, v_2)$, where a strict inequality for some set $S \subseteq N$ implies a strict inequality for the values as well.*

All of these axioms take on a natural interpretation in the QII setting. Indeed, if two features have the same probabilistic effect, no matter what other interventions are already in place, they should have the same influence. In our context, the dummy axiom says that a feature that never offers information with respect to an outcome should have no influence. Monotonicity makes intuitive sense in the QII setting: if a feature has consistently higher influence on the outcome in one setting than another, its measure of influence should increase. For example, if a user receives two transparency reports (say, for two separate loan applications), and in one report gender had a consistently higher effect on the outcome than in the other, then the transparency report should reflect this.

[147] offers an characterization of the Shapley value, based on these *monotonicity* assumption.

**Theorem 1** ([147])**.** *The Shapley value is the only function that satisfies symmetry, dummy and monotonicity.*

To conclude, the Shapley value is a *unique* way of measuring aggregate influence in the QII setting, while satisfying a set of very natural axioms.

## 2.4 Estimation

While the model we propose offers several appealing properties, it faces several technical implementation issues. Several elements of our work require significant computational effort; in particular, both the probability that a change in feature state would cause a change in outcome, and the game-theoretic influence measures are difficult to compute exactly. In the following sections we discuss these issues and our proposed solutions.

### 2.4.1 Computing Power Indices

Computing the Shapley or Banzhaf values exactly is generally computationally intractable (see [24, Chapter 4] for a general overview); however, their probabilistic nature means that they can be well-approximated via random sampling. More formally, given a random variable $X$, suppose that we are interested in estimating some determined quantity $q(X)$ (say, $q(X)$ is the mean of $X$); we say that a random variable $q^*$ is an $\varepsilon$-$\delta$ approximation of $q(X)$ if

$$\Pr[|q^* - q(X)| \geq \varepsilon] < \delta;$$

in other words, it is extremely likely that the difference between $q(X)$ and $q^*$ is no more than $\varepsilon$. An $\varepsilon$-$\delta$ *approximation scheme* for $q(X)$ is an algorithm that for any $\varepsilon, \delta \in (0, 1)$ is able to output a random variable $q^*$ that is an $\varepsilon$-$\delta$ approximation of $q(X)$, and runs in time polynomial in $\frac{1}{\varepsilon}, \log 1\delta$.

[11] show that when $\langle N, v\rangle$ is a *simple* game (i.e. a game where $v(S) \in \{0, 1\}$ for all $S \subseteq N$), there exists an $\varepsilon$-$\delta$ approximation scheme for both the Banzhaf and Shapley values; that is, for $\phi \in \{\varphi, \beta\}$, we can guarantee that for any $\varepsilon, \delta > 0$, with probability $\geq 1 - \delta$, we output a value $\phi_i^*$ such that $|\phi_i^* - \phi_i| < \varepsilon$.

More generally, [90] observe that the number of i.i.d. samples needed in order to approximate the Shapley value and Banzhaf index is parametrized in $\Delta(v) = \max_{S \subseteq N} v(S) - \min_{S \subseteq N} v(S)$. Thus, if $\Delta(v)$ is a bounded value, then an $\varepsilon$-$\delta$ approximation exists. In our setting, coalitional values are always within the interval $[0, 1]$, which immediately implies the following thm.

**Theorem 2.** *There exists an $\varepsilon$-$\delta$ approximation scheme for the Banzhaf and Shapley values in the QII setting.*

### 2.4.2 Estimating $Q$

Since we do not have access to the prior generating the data, we simply estimate it by observing the dataset itself. Recall that $\mathcal{X}$ is the set of all possible user profiles; in this case, a dataset is simply a multiset (i.e. possibly containing multiple copies of user profiles) contained in $\mathcal{X}$. Let $\mathcal{D}$ be a finite multiset of $\mathcal{X}$, the input space. We estimate probabilities by computing sums over $\mathcal{D}$. For example, for a classifier $c$, the probability of $c(X) = 1$.

$$\hat{\mathbb{E}}_{\mathcal{D}}(c(X) = 1) = \frac{\sum_{\mathbf{x} \in \mathcal{D}} 1(c(\mathbf{x}) = 1)}{|\mathcal{D}|}. \tag{2.7}$$

Given a set of features $S \subseteq N$, let $\mathcal{D}|_S$ denote the elements of $\mathcal{D}$ truncated to only the features in $S$. Then, the intervened probability can be estimated as follows:

$$\hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1) = \frac{\sum_{\mathbf{u}_S \in \mathcal{D}|_S} \sum_{\mathbf{x} \in \mathcal{D}} 1(c(\mathbf{x}|_{N \setminus S} \mathbf{u}_S) = 1)}{|\mathcal{D}|^2}. \tag{2.8}$$

Similarly, the intervened probability on individual outcomes can be estimated as follows:

$$\hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1 | X = \mathbf{x}) = \frac{\sum_{\mathbf{u}_S \in \mathcal{D}_S} 1(c(\mathbf{x}|_{N \setminus S} \mathbf{u}_S) = 1)}{|\mathcal{D}|}. \tag{2.9}$$

Finally, let us observe group disparity:

$$\left| \hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1 \mid X \in \mathcal{Y}) - \hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1 \mid X \notin \mathcal{Y}) \right|$$

The term $\hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1 \mid X \in \mathcal{Y})$ equals

$$\frac{1}{|\mathcal{Y}|} \sum_{\mathbf{x} \in \mathcal{Y}} \sum_{\mathbf{u}_S \in \mathcal{D}_S} 1(c(\mathbf{x}|_{N \setminus S} \mathbf{u}_S) = 1),$$

Thus group disparity can be written as:

$$\begin{aligned} \Big| \frac{1}{|\mathcal{Y}|} \sum_{\mathbf{x} \in \mathcal{Y}} \sum_{\mathbf{u}_S \in \mathcal{D}_S} 1(c(\mathbf{x}|_{N \setminus S} \mathbf{u}_S) = 1) \\ - \frac{1}{|\mathcal{D} \setminus \mathcal{Y}|} \sum_{\mathbf{x} \in \mathcal{D} \setminus \mathcal{Y}} \sum_{\mathbf{u}_S \in \mathcal{D}_S} 1(c(\mathbf{x}|_{N \setminus S} \mathbf{u}_S) = 1) \Big|. \end{aligned} \tag{2.10}$$

We write $\hat{Q}^{\mathcal{Y}}_{\text{disp}}(S)$ to denote (2.10).

If $\mathcal{D}$ is large, these sums cannot be computed efficiently. Therefore, we approximate the sums by sampling from the dataset $\mathcal{D}$. It is possible to show using the According to the Hoeffding bound [64], partial sums of $n$ random variables $X_i$, within a bound $\Delta$, can be well-approximated with the following probabilistic bound:

$$\Pr\left( \left| \frac{1}{n} \sum_{i=1}^{n} (X_i - \mathbb{E}\, X_i) \right| \geq \varepsilon \right) \leq 2 \exp\left( \frac{-2n\varepsilon^2}{\Delta} \right)$$

Since all the samples of measures discussed in this chapter are bounded within the interval $[0, 1]$, we admit an $\varepsilon$-$\delta$ approximation scheme where the number of samples $n$ can be chosen to be greater than $\log(2/\delta)/2\varepsilon^2$. Note that these bounds are independent of the size of the dataset. Therefore, given an efficient sampler, these quantities of interest can be approximated efficiently even for large datasets. For a fixed degree of accuracy, generating transparency reports is linear in the number of features.

## 2.5 Private Transparency Reports

One important concern is that releasing influence measures estimated from a dataset might leak information about individual users; our goal is providing accurate transparency reports, without compromising individual users' private data. To mitigate this concern, we add noise to make the measures differentially private. We show that the sensitivities of the QII measures considered in this chapter are low and therefore very little noise needs to be added to achieve differential privacy.

The *sensitivity* of a function is a key parameter in ensuring that it is differentially private; it is simply the worst-case change in its value, assuming that we change a single data point in our dataset. Given some function $f$ over datasets, we define the sensitivity of a function $f$ with respect to a dataset $\mathcal{D}$, denoted by $\Delta f(\mathcal{D})$ as

$$\max_{\mathcal{D}'} |f(\mathcal{D}) - f(\mathcal{D}')|$$

where $\mathcal{D}$ and $\mathcal{D}'$ differ by at most one instance. We use the shorthand $\Delta f$ when $\mathcal{D}$ is clear from the context.

In order to not leak information about the users used to compute the influence of an input, we use the standard Laplace Mechanism [44] and make the influence measure differentially private. The amount of noise required depends on the sensitivity of the influence measure. We show that the influence measure has low sensitivity for the individuals used to sample inputs. Further, due to a result from [71] (and stated in [83]), sampling amplifies the privacy of the computed statistic, allowing us to achieve high privacy with minimal noise addition.

The standard technique for making any function differentially private is to add Laplace noise calibrated to the sensitivity of the function:

**Theorem 3** ([44]). *For any function $f$ from datasets to $\mathbb{R}$, the mechanism $\mathcal{K}_f$ that adds independently generated noise with distribution $Lap(\Delta f(\mathcal{D})/\epsilon)$ to the $k$ output enjoys $\epsilon$-differential privacy.*

Since each of the quantities of interest aggregate over a large number of instances, the sensitivity of each function is very low.

**Theorem 4.** *Given a dataset $\mathcal{D}$,*

1. $\Delta\hat{\mathbb{E}}_{\mathcal{D}}(c(X) = 1) = \frac{1}{|\mathcal{D}|}$
2. $\Delta\hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1) \leq \frac{2}{|\mathcal{D}|}$
3. $\Delta\hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1 | X = \mathbf{x}) = \frac{1}{|\mathcal{D}|}$
4. $\hat{Q}^{\mathcal{Y}}_{disp}(S) \leq \max\left\{\frac{1}{|\mathcal{D}\cap\mathcal{Y}|}, \frac{1}{|\mathcal{D}\setminus\mathcal{Y}|}\right\}$

*Proof.* We examine some cases here. In Equation 2.7, if two datasets differ by one instance, then at most one term of the summation will differ. Since each term can only be either 0 or 1, the sensitivity of the function is

$$\Delta\hat{\mathbb{E}}_{\mathcal{D}}(c(X) = 1) = \left|\frac{0}{|\mathcal{D}|} - \frac{1}{|\mathcal{D}|}\right| = \frac{1}{|\mathcal{D}|}.$$

Similarly, in Equation 2.8, an instance appears $2|\mathcal{D}| - 1$ times, once each for the inner summation and the outer summation, and therefore, the sensitivity of the function is

$$\Delta\hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1) = \frac{2|\mathcal{D}| - 1}{|\mathcal{D}|^2} \leq \frac{2}{|\mathcal{D}|}.$$

For individual outcomes (Equation (2.9)), similarly, only one term of the summation can differ. Therefore, the sensitivity of (2.9) is $1/|\mathcal{D}|$.

Finally, we observe that a change in a single element $\mathbf{x}'$ of $\mathcal{D}$ will cause a change of at most $\frac{1}{|\mathcal{D}\cap\mathcal{Y}|}$ if $\mathbf{x}' \in \mathcal{D} \cap \mathcal{Y}$, or of at most $\frac{1}{|\mathcal{D}\setminus\mathcal{Y}|}$ if $\mathbf{x}' \in \mathcal{D} \setminus \mathcal{Y}$. Thus, the maximal change to (2.10) is at most $\max\left\{\frac{1}{|\mathcal{Y}|}, \frac{1}{|\mathcal{D}\setminus\mathcal{Y}|}\right\}$.

□

While the sensitivity of most quantities of interest is low (at most a $\frac{2}{|\mathcal{D}|}$), $\hat{Q}^{\mathcal{Y}}_{\text{disp}}(S)$ can be quite high when $|\mathcal{Y}|$ is either very small or very large. This makes intuitive sense: if $\mathcal{Y}$ is a very small minority, then any changes to its members are easily detected; similarly, if $\mathcal{Y}$ is a vast majority, then changes to protected minorities may be easily detected.

We observe that the quantities of interest which exhibit low sensitivity will have low influence sensitivity as well: for example, the local influence of $S$ is $1(c(\mathbf{x}) = 1) - \hat{\mathbb{E}}_{\mathcal{D}}(c(X_{-S}) = 1] \mid X = \mathbf{x})$; changing any $\mathbf{x}' \in \mathcal{D}$ (where $\mathbf{x}' \neq \mathbf{x}$ will result in a change of at most $\frac{1}{|\mathcal{D}|}$ to the local influence.

Finally, since the Shapley and Banzhaf indices are normalized sums of the differences of the set influence functions, we can show that if an influence function $\iota$ has sensitivity $\Delta\iota$, then the sensitivity of the indices are at most $2\Delta\iota$.

To conclude, all of the QII measures discussed above (except for group parity) have a sensitivity of $\frac{\alpha}{|\mathcal{D}|}$, with $\alpha$ being a small constant. To ensure differential privacy, we need only need add noise with a Laplacian distribution $\text{Lap}(k/|\mathcal{D}|)$ to achieve 1-differential privacy.

Further, it is known that sampling amplifies differential privacy.

**Theorem 5** ([71], [83]). *If $\mathcal{A}$ is 1-differentially private, then for any $\epsilon \in (0, 1)$, $\mathcal{A}'(\epsilon)$ is $2\epsilon$-differentially private, where $\mathcal{A}'(\epsilon)$ is obtained by sampling an $\epsilon$ fraction of inputs and then running $\mathcal{A}$ on the sample.*

Therefore, our approach of sampling instances from $\mathcal{D}$ to speed up computation has the additional benefit of ensuring that our computation is private.

Table 2.2 contains a summary of all QII measures defined in this chapter, and their sensitivity.

## 2.6 Experimental Evaluation

We illustrate the utility of the QII framework by developing three simple machine learning applications on real datasets. In Section 2.6.2, we illustrate the distinction between different quantities of interest on which Unary QII can be computed. We also illustrate the effect of discrimination on the QII measure. In Section 2.6.3, we analyze explanations for

| Name | Notation | Quantity of Interest | Sensitivity |
|---|---|---|---|
| QII on Individual Outcomes (2.1) | $\iota_{\text{ind}}(S)$ | Positive Classification of an Individual | $1/\lvert\mathcal{D}\rvert$ |
| QII on Actual Individual Outcomes (2.2) | $\iota_{\text{ind-act}}(S)$ | Actual Classification of an Individual | $1/\lvert\mathcal{D}\rvert$ |
| Average QII (2.3) | $\iota_{\text{ind-avg}}(S)$ | Average Actual Classification | $2/\lvert\mathcal{D}\rvert$ |
| QII on Group Outcomes (2.4) | $\iota^{\mathcal{Y}}_{\text{grp}}(S)$ | Positive Classification for a Group | $2/\lvert\mathcal{D}\cap\mathcal{Y}\rvert$ |
| QII on Group Disparity (2.6) | $\iota^{\mathcal{Y}}_{\text{disp}}(S)$ | Difference in classification rates among groups | $2\max(1/\lvert\mathcal{D}\setminus\mathcal{Y}\rvert, 1/\lvert\mathcal{D}\cap\mathcal{Y}\rvert)$ |

Table 2.2: A summary of the QII measures defined in this chapter

three individuals to demonstrate how Marginal QII can provide insights into individuals' classification outcomes.

We use the following datasets in our experiments:

- `adult` [84]: This standard machine learning benchmark dataset is a subset of US census data classifying individual income; it contains factors such as age, race, gender, marital status and other socio-economic parameters. We use this dataset to train a classifier that predicts the income of individuals from other parameters. Such a classifier could potentially be used to assist in credit decisions.
- `arrests` [2]: The National Longitudinal Surveys are a set of surveys conducted by the Bureau of Labor Statistics of the United States. In particular, we use the 1997 youth survey, covering young persons born in the years 1980–84.
  The survey covers various aspects of an individual's life such as medical history, criminal records and economic parameters. From this dataset, we extract the following features: age, gender, race, region, history of drug use, history of smoking, and history of arrests. We use this data to train a classifier that predicts arrests history to simulate predictive policing, where socio-economic factors are used to decide whether individuals should receive a visit from the police. A similar application is described in [68].
- `lending club`: A data set of loans originated by Lending Club [81] is used to predict charge-offs from 75 other financial variables about 890,000 individuals from which we filter 51. Charge-off prediction can be used to as a statistical basis for generating credit scores.

The three applications described above are hypothetical examples of decision making aided by algorithms that use potentially sensitive socio-economic data, and not real systems that are currently in use. We use these classifiers to illustrate the subtle causal questions that our QII measures can answer.

We use the following standard machine learning classifiers in our dataset: Logistic Regression, SVM with a radial basis function kernel, Decision Tree, and Gradient Boosted Decision Trees(see [17] for an excellent primer to these classifiers); with the exception of Logistic Regression, a linear classifier, the other three are nonlinear and can potentially learn very complex models.

### 2.6.1 Comparison with Observational Measures

In the presence of correlated inputs, observational measures often cannot identify which inputs were causally influential. To illustrate this phenomena on real datasets, we train two classifiers: (A) where gender is provided as an actual input, and (B) where gender is not provided as an input. For classifier (B), clearly the input *Gender* has no effect and any correlation between the outcome and gender is caused via inference from other inputs. In Table 2.3, for both the `adult` and the `arrests` dataset, we compute the following observational measures: Mutual Information (MI), Jaccard Index (Jaccard), Pearson Correlation (corr), and the Disparate Impact Ratio (disp) to measure the similarity between Gender and the classifiers outcome. We also measure the QII of Gender on outcome. We observe that in many scenarios the observational quantities do not change, or sometimes increase, from classifier A to classifier B, when gender is removed as an actual input to the classifier. On the other hand, if the outcome of the classifier does not depend on the input *Gender*, then the QII is guaranteed to be zero.

### 2.6.2 Unary QII Measures

In Figure 2.2, we illustrate the use of different Unary QII measures. Figures 2.2a and 2.2b show the Average QII measure (Equation 2.3) computed for features of a decision forest classifier. For the income classifier trained on the `adult` dataset, the feature with highest influence is *Marital Status*, followed by *Occupation*, *Relationship* and *Capital Gain*. Sensitive features such as *Gender* and *Race* have relatively lower influence. For the predictive policing classifier trained on the `arrests` dataset, the most influential input is *Drug History*, followed by *Gender*, and *Smoking History*. We observe that influence on outcomes may be different from influence on group disparity.

**QII on group disparity** Figures 2.2c, 2.2d show influences of features on group disparity (Equation 2.6) for two different settings. Figure 2.2c shows feature influence on group disparity by Gender in the `adult` dataset; Figure 2.2d shows the influence on group disparity by Race in the `arrests` dataset. For the income classifier trained on the `adult` dataset, we observe that most inputs have negative influence on group disparity; randomly intervening on most inputs would lead to a reduction in group disparity. In other words, a classifier that did not use these inputs would be fairer. Interestingly, in this classifier, marital status — rather than gender — has the highest influence on group disparity by gender.

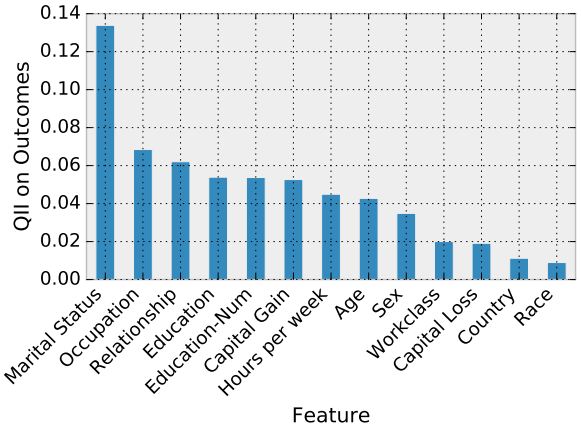
| | | logistic | | kernel svm | | decision tree | | random forest | |
|---|---|---|---|---|---|---|---|---|---|
| | | adult | arrests | adult | arrests | adult | arrests | adult | arrests |
| MI | A | 0.045 | 0.049 | 0.046 | 0.047 | 0.043 | 0.054 | 0.044 | 0.053 |
| MI | B | 0.043 | 0.050 | 0.044 | 0.053 | 0.042 | 0.051 | 0.043 | 0.052 |
| Jaccard | A | 0.501 | 0.619 | 0.500 | 0.612 | 0.501 | 0.614 | 0.501 | 0.620 |
| Jaccard | B | 0.500 | 0.611 | 0.501 | 0.615 | 0.500 | 0.614 | 0.501 | 0.617 |
| corr | A | 0.218 | 0.265 | 0.220 | 0.247 | 0.213 | 0.262 | 0.218 | 0.262 |
| corr | B | 0.215 | 0.253 | 0.218 | 0.260 | 0.215 | 0.257 | 0.215 | 0.259 |
| disp | A | 0.286 | 0.298 | 0.377 | 0.033 | 0.302 | 0.335 | 0.315 | 0.223 |
| disp | B | 0.295 | 0.301 | 0.312 | 0.096 | 0.377 | 0.228 | 0.302 | 0.129 |
| QII | A | 0.036 | 0.135 | 0.044 | 0.149 | 0.023 | 0.116 | 0.012 | 0.109 |
| QII | B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 2.3: Comparison of QII with associative measures. For 4 different classifiers, we compute metrics such as Mutual Information (MI), Jaccard Index (JI), Pearson Correlation (corr), Group Disparity (disp) and Average QII between Gender and the outcome of the learned classifier. Each metric is computed in two situations: (A) when Gender is provided as an input to the classifier, and (B) when Gender is not provided as an input to the classifier.

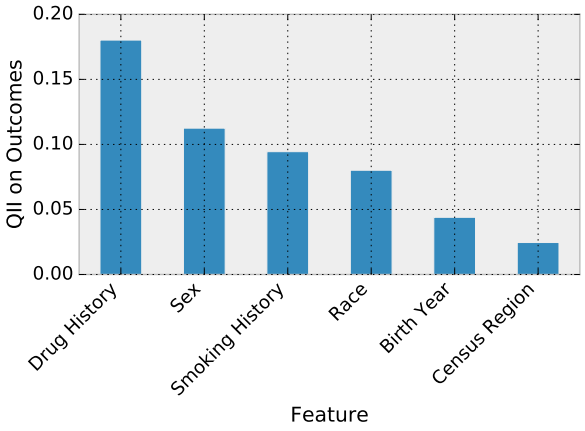
For the `arrests` dataset, most inputs have the effect of increasing group disparity if randomly intervened on. In particular, *Drug history* has the highest positive influence on disparity in `arrests`. Although Drug history is correlated with race, using it reduces disparate impact by race, i.e. makes fairer decisions.

In both examples, features correlated with the sensitive attribute are the most influential for group disparity according to the sensitive attribute rather than the sensitive attribute itself. It is in this sense that QII measures can identify proxy variables that cause associations between outcomes and sensitive attributes.

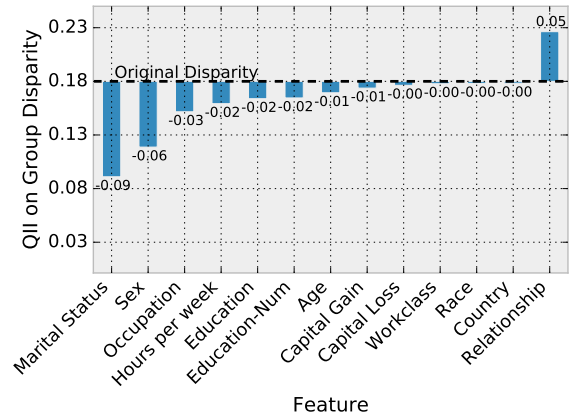
**QII with artificial discrimination** We simulate discrimination using an artificial experiment. We first randomly assign "ZIP codes" to individuals in our dataset (these are simply arbitrary numerical identifiers). Then, to simulate systematic bias, we make an $f$ fraction of the ZIP codes discriminatory in the following sense: all individuals in the protected set are automatically assigned a negative classification outcome. We then study the change in the influence of features as we increase $f$. Figure 2.3a, shows that the influence of *Gender* increases almost linearly with $f$. Recall that *Marital Status* was the most influential feature for this classifier without any added discrimination. As $f$ increases, the importance of *Marital Status* decreases as expected, since the number of individuals for



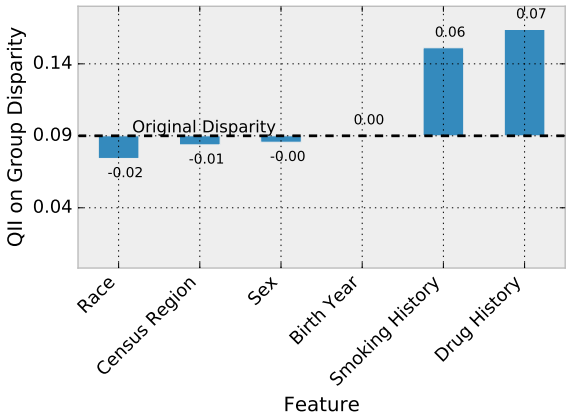

(a) QII of inputs on Outcomes for the `adult` dataset




(b) QII of inputs on Outcomes for the `arrests` dataset

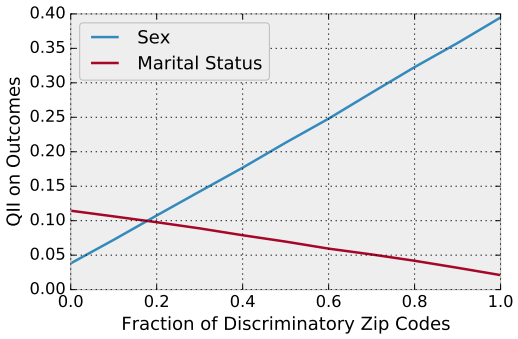



(c) QII of Inputs on Group Disparity by Gender in the `adult` dataset

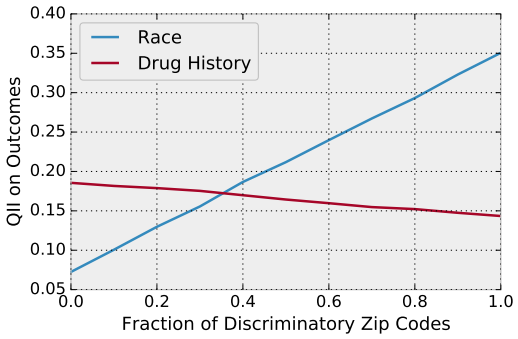



(d) Influence on Group Disparity by Race in the `arrests` dataset

Figure 2.2: QII measures for the `adult` and `arrests` datasets




(a) Change in QII of inputs as discrimination by Zip Code increases in the `adult` dataset




(b) Change in QII of inputs as discrimination by Zip Code increases in the `arrests` dataset

Figure 2.3: The effect of discrimination on QII.

| | |
|---|---|
| Age | 23 |
| Workclass | Private |
| Education | 11th |
| Education-Num | 7 |
| Marital Status | Never-married |
| Occupation | Craft-repair |
| Relationship | Own-child |
| Race | Asian-Pac-Islander |
| Gender | Male |
| Capital Gain | 14344 |
| Capital Loss | 0 |
| Hours per week | 40 |
| Country | Vietnam |

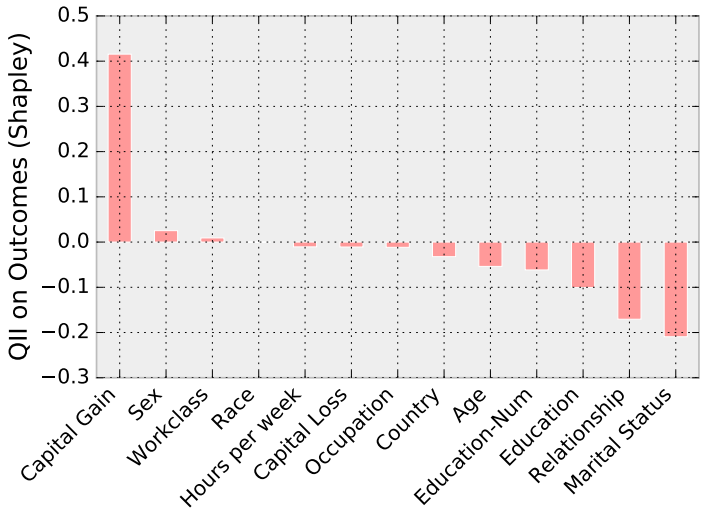



Figure 2.4: Mr. X's profile and explanation for negative classification.

whom *Marital Status* is pivotal decreases.

### 2.6.3 Personalized Explanations

To illustrate the utility of personalized explanations, we study the classification of individuals who received potentially unexpected outcomes. For the personalized explanations, we use classification outcomes obtained from decision forests, though one obtains a similar outcome using other classifiers.

The influence measure that we employ is the Shapley value. In more detail, the coalitional function we use is $v(S) = \iota^{Q_{\mathcal{A}}}(S)$, with $Q_{\mathcal{A}}$ being $\mathbb{E}[c(\cdot) = 1 \mid X = \mathbf{x}]$; that is, the marginal contribution of $i \in N$ to $S$ is given by $m_i(S) = \mathbb{E}[c(X_{-S}) = 1 \mid X = \mathbf{x}] - \mathbb{E}[c(X_{-S\cup\{i\}}) = 1 \mid X = \mathbf{x}]$.

We emphasize that some features may have a negative Shapley value; this should be interpreted as follows: a feature with a high positive Shapley value often increases the certainty that the classification outcome is 1, whereas a feature whose Shapley value is negative is one that increases the certainty that the classification outcome would be zero.

**Mr. X:** The first example is of an individual from the `adult` dataset, to whom we refer as Mr. X (Figure 2.4). The learnt classifier classifies his income as low. This result may be surprising to him: he reports high capital gains ($14k), and only 2.1% of people with capital gains higher than $10k are reported as low income. In fact, he might be led to believe that his classification may be a result of his ethnicity or country of origin. Examining his explanation in Figure 2.4, however, we find that the most influential features that led to his negative classification were Marital Status, Relationship and Education.

| | |
|---|---|
| Age | 27 |
| Workclass | Private |
| Education | Preschool |
| Education-Num | 1 |
| Marital Status | Married-civ-spouse |
| Occupation | Farming-fishing |
| Relationship | Other-relative |
| Race | White |
| Gender | Male |
| Capital Gain | 41310 |
| Capital Loss | 0 |
| Hours per week | 24 |
| Country | Mexico |

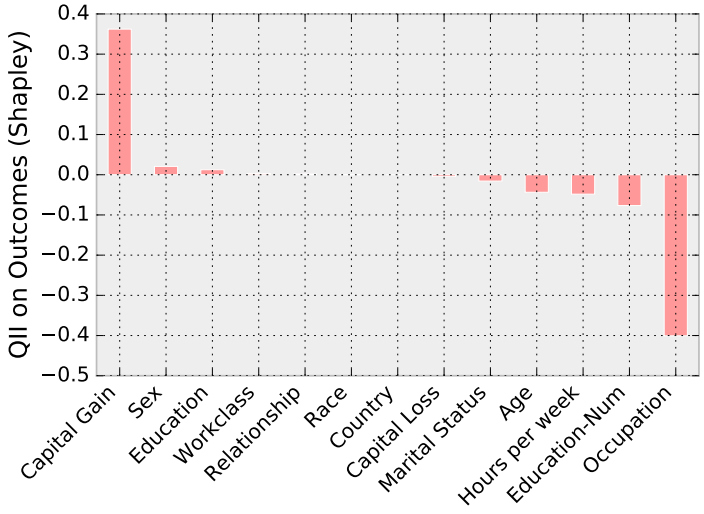



Figure 2.5: Mr. Y's profile and explanation for negative classification.

**Mr. Y:** The second example, to whom we refer as Mr. Y (Figure 2.5), has even higher capital gains than Mr. X. Mr. Y is a 27 year old, with only Preschool education, and is engaged in fishing. Examination of the transparency report reveals that the most influential factor for negative classification for Mr. Y is his Occupation. Interestingly, his low level of education is not considered very important by this classifier.

**Mr. Z:** The third example, who we refer to as Mr. Z (Figure 2.6) is from the `arrests` dataset. History of drug use and smoking are both strong indicators of arrests. However, Mr. X received positive classification by this classifier even without any history of drug use or smoking. Upon examining his classifier, it appears that race, age and gender were most influential in determining his outcome. In other words, the classifier that we train for this dataset (a decision forest) has picked up on the correlations between race (Black), and age (born in 1984) to infer that this individual is likely to engage in criminal activity. Indeed, our interventional approach indicates that this is not a mere correlation effect: race is actively being used by this classifier to determine outcomes. Of course, in this instance, we have explicitly offered the race parameter to our classifier as a viable feature. However, our influence measure is able to pick up on this fact, and alert us of the problematic behavior of the underlying classifier. More generally, this example illustrates a concern with the black box use of machine learning which can lead to unfavorable outcomes for individuals.

### 2.6.4 Differential Privacy

Most QII measures considered in this chapter have low sensitivity, and therefore can be made differentially private with negligible loss in utility. However, recall that the sensitivity of influence measure on group disparity $\iota^{\mathcal{Y}}_{\text{disp}}$ depends on the size of the protected group in

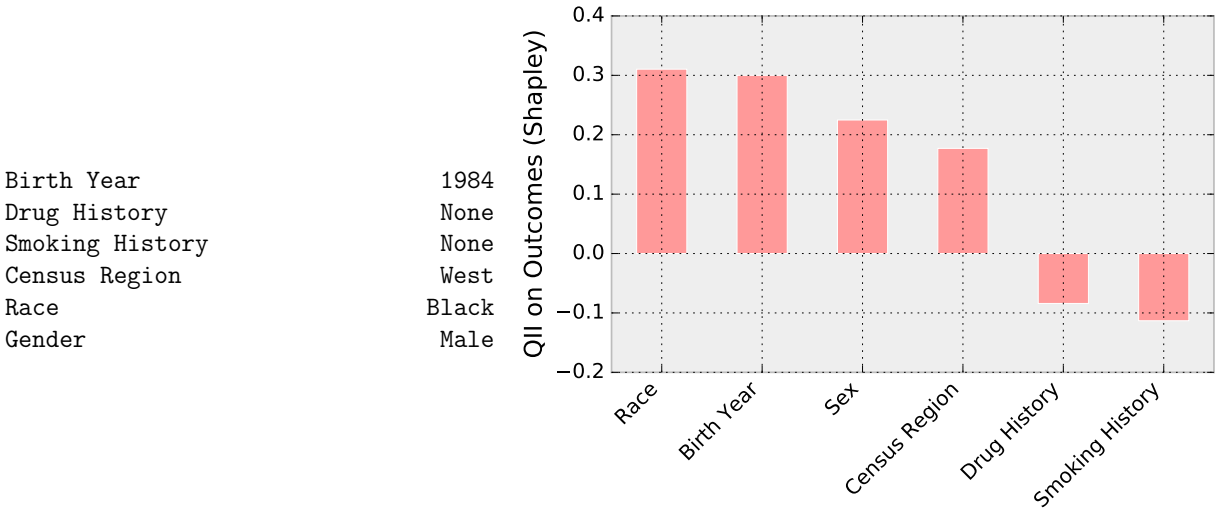



Figure 2.6: Mr. Z's profile and explanation for positive classification.

the dataset $\mathcal{D}$ as follows:

$$\iota^{\mathcal{Y}}_{\text{disp}} = 2\max\left(\frac{1}{|\mathcal{D} \setminus \mathcal{Y}|}, \frac{1}{|\mathcal{D} \cap \mathcal{Y}|}\right)$$

For sufficiently small minority groups, a large amount of noise might be required to ensure differential privacy, leading to a loss in utility of the QII measure. To estimate the loss in utility, we set a noise of 0.005 as the threshold of noise at which the measure is no longer useful, and then compute fraction of times noise crosses that threshold when Laplacian noise is added at $\epsilon = 1$. The results of this experiment are as follows:

| $\mathcal{Y}$ | Count | Loss in Utility |
|---|---|---|
| Race: White | 27816 | $2.97 \times 10^{-14}$ |
| Race: Black | 3124 | $5.41 \times 10^{-14}$ |
| Race: Asian-Pac-Islander | 1039 | $6.14 \times 10^{-05}$ |
| Race: Amer-Indian-Eskimo | 311 | 0.08 |
| Race: Other | 271 | 0.13 |
| Gender: Male | 21790 | $3.3 \times 10^{-47}$ |
| Gender: Female | 10771 | $3.3 \times 10^{-47}$ |

We note that for most reasonably sized groups, the loss in utility is negligible. However, the Asian-Pac-Islander, and the Amer-Indian-Eskimo racial groups are underrepresented in this dataset. For these groups, the QII on Group Disparity estimate needs to be very noisy to protect privacy.

| | logistic | kernel-svm | decision-tree | decision-forest |
|---|---|---|---|---|
| QII on Group Disparity | 0.56 | 234.93 | 0.57 | 0.73 |
| Average QII | 0.85 | 322.82 | 0.77 | 1.12 |
| QII on Individual Outcomes (Shapley) | 6.85 | 2522.3 | 7.78 | 9.30 |
| QII on Individual Outcomes (Banzhaf) | 6.77 | 2413.3 | 7.64 | 10.34 |

Table 2.4: Runtimes in seconds for explanation computation

### 2.6.5 Performance

We report runtimes of our prototype for generating explanations on the `adult` dataset. Recall from Section 2.5 that we approximate QII measures by computing sums over samples of the dataset. According to the Hoeffding bound to derive an $(\epsilon, \delta)$ estimate of a QII measure, at $\epsilon = 0.01$, and $n = 37000$ samples, $\delta = 2\exp(-n\epsilon^2) < 0.05$ is an upper bound on the probability of the output being off by $\epsilon$. Table 2.4 shows the runtimes of four different QII computations, for 37000 samples each. The runtimes of all algorithms except for kernel SVM are fast enough to allow real-time feedback for machine learning application developers. Evaluating QII metrics for Kernel SVMs is much slower than the other metrics because each call to the SVM classifier is very computationally intensive due to a large number of distance computations that it entails. We expect that these runtimes can be optimized significantly. We present them as proof of tractability.

Runtimes on the `lending club` dataset scale roughly linearly to that of the adult dataset with QII on Individual Outcomes (Shapley) requiring 29.6 seconds on average with 51 input features, and 37000 samples (same as `adult`) out of 890000 instances. As the accuracy does not depend on the size of the dataset sampled from, this achieves similar theoretical accuracy to the `adult` dataset.

### 2.6.6 Conciseness

Regulations surrounding credit such as the Equal Credit Opportunity Act [4] require concise explanations for the principal reasons behind denials. We evaluate QII explanations in terms of the number of important reasons revealed for individual outcomes. For a 500 tree random forest trained on the `lending club` dataset with 51 input features, we computed the number of negative influencers, that is, the number of features that have negative influence with magnitude greater than 0.001. The average number of negative influencers were 4.6 indicating that a large fraction of the influence is concentrated in a small number of features for any individual.

## 2.7 QII for Fairness

Due to the widespread and black box use of machine learning in aiding decision making, there is a legitimate concern of algorithms introducing and perpetuating social harms such as racial discrimination [13, 106]. As a result, the algorithmic foundations of fairness in personal information processing systems have received significant attention recently [22, 32, 45, 70, 149]. While many of the algorithmic approaches [22, 50, 70, 149] have focused on group parity as a metric for achieving fairness in classification, Dwork et al. [45] argue that group parity is insufficient as a basis for fairness, and propose an approach which prescribes that similar individuals should receive similar classification outcomes. However, this approach requires a similarity metric for individuals which is often subjective and difficult to construct.

QII does not suggest any normative definition of fairness. Instead, we view QII as a diagnostic tool to aid fine-grained fairness determinations. In fact, QII can be used in the spirit of the similarity based definition of [45] by comparing the personalized privacy reports of individuals who are *perceived* to be similar but received different classification outcomes, and identifying the inputs which were used by the classifier to provide different outcomes.

When group parity is used as a criteria for fairness, QII can identify the features that lead to group disparity, thereby identifying features being used by a classifier as a proxy for sensitive attributes. The determination of whether using certain proxies for sensitive attributes is discriminatory is often a task-specific normative judgment. For example, using standardized test scores (e.g., SAT scores) for college admissions decisions is by and large accepted. However, SAT scores may act as a proxy for several protected attributes, leading to concerns of unfairness [132, 140]. Our goal is not to provide such normative judgments. Rather we seek to provide fine-grained transparency into input usage (e.g., what's the extent to which SAT scores influence decisions), which is useful to make determinations of discrimination from a specific normative position.

## 2.8 Related Work

QII enables algorithmic transparency via a family of game-theoretic causal influence measures. In this section we compare with related work along a number of different dimensions. First, we compare with related approaches to measuring causal influence, associations, and briefly discuss an emerging direction that measures proxy influence via a combination of associative and causal techniques. Next, we discuss orthogonal approaches to algorithmic transparency: interpretable machine learning and experimentation on web services. Finally, we discuss other fields in which similar game-theoretic measures have been applied.

**Quantitative Causal Measures** Causal models and probabilistic interventions have been used in a few other settings. While the form of the interventions in some of these settings may be very similar, our generalization to account for different quantities of interests enables us to reason about a large class of transparency queries for data analytics systems

ranging from classification outcomes of individuals to disparity among groups. Further, the notion of marginal contribution which we use to compute responsibility does not appear in this line of prior work.

Janzing et al. [67] use interventions to assess the causal importance of relations between variables in causal graphs; in order to assess the causal effect of a relation between two variables, $X \to Y$ (assuming that both take on specific values $X = x$ and $Y = y$), a new causal model is constructed, where the value of $X$ is replaced with a prior over the possible values of $X$. The influence of the causal relation is defined as the distance between the joint distributions of all the variables in the two causal models with and without the value of $X$ replaced. The approach of intervening with a random value from the prior is similar to our approach of constructing $X_{-S}U_S$.

Independently, there has been considerable work in the machine learning community to define importance metrics for variables, mainly for the purpose of feature selection (see [59] for a comprehensive overview). One important metric is called Permutation Importance [19], which measures the importance of a feature towards classification by randomly permuting the values of the feature and then computing the difference of classification accuracies before and after the permutation. Replacing a feature with a random permutation can be viewed as sampling the feature independently from the prior.

**Measures of Association** One can think of our results as a causal alternative to *quantitative information flow*. Quantitative information flow is a broad class of metrics that quantify the information leaked by a process by comparing the *information* contained before and after observing the outcome of the process. Quantitative Information Flow traces its information-theoretic roots to the work of Shannon [118] and Rényi [110]. Recent works have proposed measures for quantifying the security of information by measuring the amount of information leaked from inputs to outputs by certain variables; we point the reader to [123] for an overview, and to [29] for an exposition on information theory. Quantitative Information Flow is concerned with information leaks and therefore needs to account for correlations between inputs that may lead to leakage. The dual problem of transparency, on the other hand, requires us to destroy correlations while analyzing the outcomes of a system to identify the causal paths for information leakage. Measures of association have also been widely used for detecting discrimination (see Section 6.9).

**Proxy Influence** An emerging direction in this space is the identification of proxy or indirect use of sensitive features as opposed to direct causal use, as captured by QII. Adler et al. [7] quantify the indirect influence of an attribute by obscuring the attribute (along with associations) from a dataset and comparing the prediction accuracy of a model before and after obscuring. In a different approach, Datta et al. [**?** ] identify intermediate computations in a program that are associated with an attribute and use QII to measure the causal influence of the intermediate computations on the outcome.

**Interpretable Machine Learning** An orthogonal approach to adding interpretability to machine learning is to constrain the choice of models to those that are interpretable by

design. This can either proceed through regularization techniques such as Lasso [133] that attempt to pick a small subset of the most important features, or by using models that structurally match human reasoning such as Bayesian Rule Lists [82], Supersparse Linear Integer Models [144], or Probabilistic Scaling [113]. Since the choice of models in this approach is restricted, a loss in predictive accuracy is a concern, and therefore, the central focus in this line of work is the minimization of the loss in accuracy while maintaining interpretability. On the other hand, our approach to interpretability is forensic. We add interpretability to machine learning models after they have been learnt. As a result, our approach does not constrain the choice of models that can be used.

**Experimentation on Web Services** Another emerging body of work focuses on systematic experimentation to enhance transparency into Web services such as targeted advertising [12, 32, 57, 79, 80]. The setting in this line of work is different since they have restricted access to the analytics systems through publicly available interfaces. As a result they only have partial control of inputs, partial observability of outputs, and little or no knowledge of input distributions. The intended use of these experiments is to enable external oversight into Web services without any cooperation. Our framework is more appropriate for a transparency mechanism where an entity proactively publishes transparency reports for individuals and groups. Our framework is also appropriate for use as an internal or external oversight tool with access to mechanisms with control and knowledge of input distributions, thereby forming a basis for testing.

**Game-Theoretic Influence Measures** Recent years have seen game-theoretic influence measures used in various settings. Datta et al. [31] also define a measure for quantifying feature influence in classification tasks. Their measure does not account for the prior on the data, nor does it use interventions that break correlations between sets of features. In the terminology of this chapter, the quantity of interest used by [31] is the ability of changing the outcome by changing the state of a feature. Strumbelj and Kononenko [128] also use the Shapley value to compute influences for individual classification. This work greatly extends and generalizes the concepts presented in [31] and [128], by both accounting for interventions on sets, and by generalizing the notion of influence to include a wide range of system behaviors, such as group disparity, group outcomes and individual outcomes. Essentially, the instantiations of our transparency schema define a wide range of transparency reports.

Game theoretic measures have been used by various research disciplines to measure influence. Indeed, such measures are relevant whenever one is interested in measuring the marginal contribution of variables, and when sets of variables are able to cause some measurable effect. Prominent domains where game theoretic measures have recently been used are terrorist networks [86, 92], protein interactions [18], and neurophysical models [72]. The novelty in our use of the game theoretic power indices lies in the conception of a cooperative game via a valuation function $\iota(S)$, defined by an randomized intervention on inputs $S$. Such an intervention breaks correlations and allows us to compute marginal causal influences on a wide range of system behaviors.

## 2.9 Conclusion

In this chapter, we present QII, a general family of metrics for quantifying the influence of inputs in systems that process personal information. In particular, QII lends insights into the behavior of opaque machine learning algorithms by allowing us to answer a wide class of queries ranging from influence on individual causal outcomes to influence on disparate impact. To achieve this, QII breaks correlations between inputs to allow causal reasoning, and computes the marginal influence of inputs in situations where inputs cannot affect outcomes alone. Also, we demonstrate that QII can be efficiently approximated, and can be made differentially private with negligible noise addition in many cases.

A problem which we don't address in this chapter is that the kind of interventions presented here my query the model on instances outside the natural distribution of instances. We explore two solutions to this problem: (i) we train the model on these intervened instances via an active learning approach in Chapter 3, and (ii) for differentiable models, we explore infinitesimal interventions in Chapter 4. Also, we do not consider the problem of indirect use: identifying the influence of inputs that are not explicitly provided but inferred by the model. We address this problem in Chapter 6. Finally, we do not consider situations where inputs do not have well understood semantics. Such situations arise often in settings such as image or speech recognition, and automated video surveillance. With the proliferation of immense processing power, complex machine learning models such as deep neural networks have become ubiquitous in these domains. We consider this problem in Chapter 5.

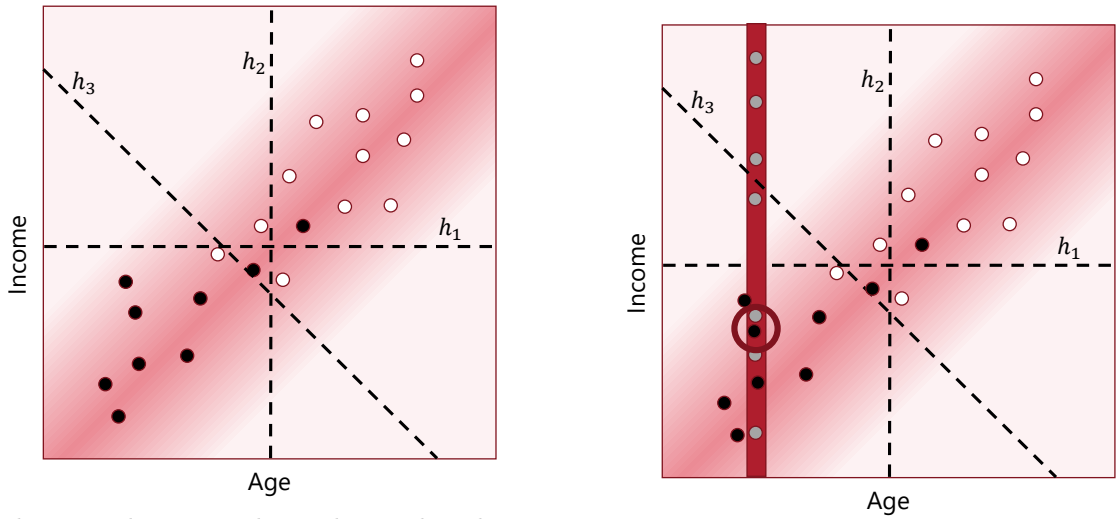
# Chapter 3

# Supervising Input Influence

In Chapter 2, we present an approach to addressing the converged use problem by proposing a causal influence measure that quantifies the relative importance of inputs used by a model. However, causal analyses, including QII, involve evaluating the classifier using datapoints that may be atypical of its training distribution. In this chapter, we show that standard methods for training classifiers that minimize empirical risk do not constrain the behavior of the classifier on such datapoints. As a result, training to minimize empirical risk does not distinguish among classifiers that agree on predictions in the training distribution but have wildly different causal influences. We term this problem *covariate shift in causal testing* and formally characterize conditions under which it arises. As a solution to this problem, we propose a novel active learning algorithm that constrains the influence measures of the trained model. We prove that any two predictors whose errors are close on both the original training distribution and the distribution of atypical points are guaranteed to have causal influences that are also close. Further, we empirically demonstrate with synthetic labelers that our algorithm trains models that (i) have similar causal influences as the labeler's model, and (ii) generalize better to out-of-distribution points while (iii) retaining their accuracy on in-distribution points.

Data processors employing machine learning algorithms are increasingly being required to provide and account for reasons behind their predictions due to regulations such as the EU GDPR [46]. This call for reasoning tools has intensified with the increasing use of machine learning systems in domains like criminal justice [8], credit [21], and hiring [66]. Traditionally, influence measures were used to inform feature selection [19]. Recently, influence measures have received renewed interest as part of a toolbox to explain operations and reveal biases of inscrutable machine learning systems [7, 34, 36, 74, 111].

Causal influence measures are a particularly important constituent of this toolbox [34, 74]. In Chapter 2, we demonstrated how causal influence measures enable identification of principal reasons for decisions (e.g., credit denials) by evaluating *counterfactual queries* that ask whether changing input attributes would produce a change in the decision. This determination is used to explain and guard against unjust biases. For example, the use of a correlate of age like income to make credit decisions may be justified even if it causes applicants of one age group to be approved at a higher rate than another whereas the direct




(a) Three predictors with similar in-distribution predictions

(b) Causal testing for income

Figure 3.1: The three predictors trying to separate white points from black have similar predictions on the distribution, but very different causal influences. Predictor $h_1$ uses only income, $h_2$ uses only age, and $h_3$ uses a linear combination of the two. Figure (b) shows how counterfactual querying, by keeping age fixed and varying income allows the identification of causal influence and distinguishes between the three classifiers.

use of age or a correlate like zipcode may not be justified[1].

Causal analyses of natural systems often involve observing outcomes of specially created units, e.g. mice with genes altered. Such units may be atypical in natural populations. However, while performing causal analysis over machine learnt systems, a similar approach encounters an important challenge: machine learning systems are not expected to be evaluated on atypical or out-of-distribution units (datapoints), since they have not been exposed to such units during training. Standard methods for training classifiers that minimize empirical risk do not constrain the behavior of the classifier on such datapoints. As a result, training to minimize empirical risk does not distinguish among classifiers that agree on predictions in the training distribution but have unintended causal influences. We term this problem *covariate shift in causal testing*. In other words, typical machine learning algorithms are designed to make the right predictions but not necessarily for justifiable reasons.

Returning to the example of credit decisions using age and income, consider a situation where the two are strongly correlated: young individuals have low income and older individuals have a higher income. This situation is illustrated in Figure 3.1a where all three predictors $h_1, h_2, h_3$ have low predictive error, but they make similar predictions for very different reasons. Since the three predictors have nearly identical predictions on the distribution, points from the distribution are not useful in distinguishing the causal influ-

[1]This is an example of a "business necessity defense" under US law on disparate impact [20].

ence of the two features. As a result, causal testing requires the creation of atypical units that break the correlations between features. For example, evaluating the predictor on the points on the red bar (Figure 3.1b) where age is fixed and income is varied informs whether income is used by a given predictor or not. However, since from an empirical risk minimization perspective the atypical points are irrelevant, an algorithm optimizing just for predictive accuracy is free to choose any of the three predictors.

We formally characterize conditions that give rise to covariate shift in causal testing (Theorem 6). Intuitively, this result states that if the units used for measuring causal influence are sufficiently outside the data distribution, constraining the behavior of a predictor on the data distribution does not constrain the causal influences of the predictor.

In order to address this issue, we introduce an active learning algorithm in Section 3.3. This algorithm provides an accountability mechanism for data processors to examine important features, and if their influences are suspicious, to collect additional information that constrains the feature influences. This additional information could steer the influences toward more acceptable values (e.g., by reducing the influence of age in $h_2$ in Figure 3.1a). Alternatively, it could provide additional evidence that the influence values convey appropriate predictive power and the suspicions are unfounded (e.g., by preserving influences in $h_1$ in Figure 3.1a).

The active learning process is assisted by two oracles. The first is a feature selection oracle that examines the causal influences of different features, and chooses the feature for which counterfactuals queries should be answered. We envision this oracle to be an auditor who can identify problematic causal influences based on background knowledge of causal factors or ethical norms governing classification. The second is similar to a standard active learning oracle, and labels atypical points to answer counterfactual queries. For example, for predictor $h_2$ in our running example, the feature selection oracle might notice that age has an unduly high influence, and can instruct the algorithm to focus on instances that vary age while keeping income fixed. While the direct use of age may be obviously problematic, in common applications the system designer may not have apriori knowledge of which attribute uses are problematic. The feature selection oracle may be able to spot suspiciously high or low influences and guide the counterfactual queries that get sent to the labeler to better inform the learning.

We evaluate the counterfactual active learning algorithm for linear, decision tree, and random forest models on a number of datasets, using a synthetic labeler. In particular, we demonstrate that after counterfactual active learning, the trained classifier has similar causal influence measures to the labeler. We also show that the classifier can generalize better to out-of-distribution points. This is an important consequence of having causal behavior similar to the labeler. Finally, we demonstrate that the accuracy on the data distribution does not degrade as a result of this additional training.

**Related Work.** Prior work on causal learning learns the structure of causal models [65, 126], or given the structure of models, the functional relationship between variables. In this context, active learning has been used to aid both the discovery of causal structures [63, 134] and their functional relationships [112]. In this work we don't attempt to learn true causal

models. Instead, our work focuses on constraining the causal behavior of learnt models. In doing so, we provide an accountability mechanism for data processors to collect additional data that guides the causal influences of their models to more acceptable values or justifies the causal influences of the learnt model.

**Contributions.** In summary, the contributions of this chapter are as follows.

- A formal articulation of the *covariate shift in causal testing* problem.
- A novel active learning algorithm that addresses the problem.
- An empirical evaluation of the algorithm for standard machine learning predictors on a number of real-world datasets.

## 3.1 Background

A predictor $h$ is a function $\mathcal{X} \to \mathcal{Y}$ that operates on an input space $\mathcal{X} \subseteq \mathbb{R}^n$ to a space of predictions $\mathcal{Y}$. The input space $\mathcal{X}$ has a probability distribution $P$ associated with it, where $P(X = x)$ is the frequency of drawing a particular instance $x$.

### 3.1.1 Risk Minimization

Given random variables $X \in \mathcal{X}$, and $Y \in \mathcal{Y}$, and a loss function $l$, the *risk* associated with predictor $h$ is given by

$$R(h) = \mathbb{E}\left[l(h(X), Y)\right].$$

The goal of supervised learning algorithms under a risk minimization paradigm is to minimize $R(h)$. In general, the distributions over $X$ and $Y$ are unknown. As a result, learning algorithms minimize *empirical risk* over a sample $\{(x_i, y_i)\}_{1..N}$

$$\hat{R}(h) = \frac{1}{N}\sum_{i=1}^{N} l(h(x_i), y_i).$$

Note that the risk minimization paradigm only constrains the behavior of a predictor on points from the distribution and treats any two predictors that have identical behavior on points from the distribution interchangeably.

For ease of presentation, we focus on binary classification tasks where $\mathcal{Y}$ is binary, and use the $0-1$ loss function $l_{0\text{-}1}(\hat{y}, y) = I(\hat{y} \neq y) = |\hat{y} - y|$.

### 3.1.2 Counterfactual Influence

The influence of a feature $f$ for a predictor $h$ is measured by comparing the outcomes of $h$ on the data distribution to the outcomes of a counterfactual distribution that changes the value of $f$. We denote the data distribution over features as $X$ and the counterfactual distribution with respect to feature $f$ as $X^f_{\text{cf}}$.

A number of influence measures proposed in prior work can be viewed as instances of this general idea. For example, Permutation Importance [19], measures the difference

in accuracy between $X$ and $X^f_{\text{cf}}$, where $X^f_{\text{cf}}$ is chosen as $X$ randomly permuted. In [7], $X^f_{\text{cf}}$ is chosen as the minimal perturbation of $X$ such that feature $f$ cannot be predicted. In this chapter, we use Average Unary QII (auQII), an instance of Quantitative Input Influence [34], as our causal influence measure. The counterfactual distribution $X^f_{\text{cf}}$ for auQII is represented as $X_{-f}U_f$, where the random variable $X_{-f}$ represents features except $f$ where $U_f$ is sampled from the marginal distribution of $f$ independently of the rest of the features $X_{-f}$.

$$P(X_{-f}U_f = x) = P(X_{-f} = x_{-f})P(U_f = x_f)$$

**Definition 7.** *Given a loss function $l$, and a model $h$, the Average Unary QII (auQII) of an input $f$, written $\iota_f(h)$, is defined as*

$$\iota_f(h) = \mathop{\mathbb{E}}_{X,U_f}[l_{0\text{-}1}(h(X), h(X_{-f}U_f))]$$

## 3.2 Covariate shift in Causal Testing

In this section, we discuss some of the theoretical implications of the covariate shift in causal testing. First, we show in Theorem 6 that risk minimization does not constrain influences when the data distribution diverges significantly from the counterfactual distribution. In other words, predictors trained under an ERM regime are free to choose influential factors. Further, in Theorem 7, we demonstrate predictors that agree on predictions on both the data distribution and the counterfactual distribution have similar influences. This theorem forms the motivation for our counterfactual active learning algorithm presented in Section 3.3 that attempts to minimize errors on both the data and the counterfactual distribution by adding points from the counterfactual distribution to the training set.

### 3.2.1 Counterfactual divergence

We first define what it means for an influence measure to be unconstrained by its behavior on the data distribution. An influence measure $\iota_f$ is unconstrained for a predictor $h$, and data distribution $X$, if it is possible to find a predictor $h'$ which has similar predictions on the data distribution but very different influences. More specifically, if the influence is high, then it can be reduced to a lower value, and vice versa.

**Definition 8.** *An influence measure $\iota_f$ is said to be $(\epsilon, \delta)$-unconstrained, for $0 \leq \delta \leq 1/2$, for a predictor $h$, if there exists predictors $h_1, h_2$ such that for $i \in \{1, 2\}$, $P(h(X) \neq h_i(X)) \leq \epsilon$, and $\iota_f(h_1) \geq \delta$ and $\iota_f(h_2) \leq 1 - \delta$.*

The following theorem shows that if there exist regions $\varphi$ in the input space with low probability weight in the data distribution and high weight in the counterfactual distribution, i.e. the data distribution and counterfactual distribution diverge significantly, then any model will have unconstrained causal influences. As a result, predictors trained under an ERM regime are free to choose influential causal factors.

**Theorem 6.** *If there exists a predicate $\varphi$, such that $P(\varphi(X)) \leq \epsilon$ and $P(\varphi(X_{-f}U_f) \wedge \neg\varphi(X)) = \gamma$, then for any $h$, $\iota_f(h)$ is $(\epsilon, \gamma/2)$-unconstrained.*

*Proof.* The proof proceeds via an averaging argument. Let $\Pi$ be the set of all functions from $\mathcal{X}$ to $\{0,1\}$. For $i \in \{1,2\}$ consider $h_i$ sampled uniformly from the set of deterministic functions that map values $x$ satisfying $\varphi$ to $h(x)$ and according to some $\pi \in \Pi$ otherwise: $\{x \mapsto h(x) \text{ when } \neg\varphi(x) \text{ and } \pi(x) \text{ o.w. }\}_{\pi\in\Pi}$. Notice that $P(h_i(X)|\neg\varphi(X))$ is therefore uniform in $\{0,1\}$.

As $h_i(x) = h(x)$ when $\neg\varphi(x)$, any such classifier satisfies $P(h(X) \neq h_i(X)) \leq P(\varphi(x)) \leq \epsilon$. Computing the expected influence over all $h_1$, we have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{h_1}[\iota_f(h_1))] \\
=&\mathop{\mathbb{E}}_{h_1}\left[\mathop{\mathbb{E}}_{X,U_f}[I(h_1(X) \neq h_1(X_{-f}U_f))]\right] \\
&\text{Let } \theta = \varphi(X_{-f}U_f) \wedge \neg\varphi(X). \text{ Then, } P(\theta) = \gamma \\
=&\gamma \mathop{\mathbb{E}}_{X,U_f}\left[\mathop{\mathbb{E}}_{h_1}[I(h_1(X) \neq h_1(X_{-f}U_f))] \mid \theta\right] \\
&+ (1-\gamma)\mathop{\mathbb{E}}_{X,U_f}\left[\mathop{\mathbb{E}}_{h_1}[I(h_1(X) \neq h_1(X_{-f}U_f))] \mid \neg\theta\right] \\
&\left(\text{if } \theta, \text{ then } \mathop{\mathbb{E}}_{h_1}[I(h_1(X) \neq h_1(X_{-f}U_f))] = \frac{1}{2}\right) \\
\geq&\gamma\frac{1}{2} + (1-\gamma)0 \\
=&\gamma/2.
\end{aligned}
$$

By an averaging argument, there exists an $h_1^*$ such that $\iota_f(h_1^*) \geq \gamma/2$. Similarly, computing the expected influence over all $h_2$, we have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{h_2}[\iota_f(h_2))] \\
=&\mathop{\mathbb{E}}_{h_2}\left[\mathop{\mathbb{E}}_{X,U_f}[I(h_2(X) \neq h_2(X_{-f}U_f))]\right] \\
&\text{Let } \theta = \varphi(X_{-f}U_f) \wedge \neg\varphi(X). \text{ Then, } P(\theta) = \gamma \\
=&\gamma \mathop{\mathbb{E}}_{X,U_f}\left[\mathop{\mathbb{E}}_{h_2}[I(h_2(X) \neq h_2(X_{-f}U_f))] \mid \theta\right] \\
&+ (1-\gamma)\mathop{\mathbb{E}}_{X,U_f}\left[\mathop{\mathbb{E}}_{h_2}[I(h_2(X) \neq h_2(X_{-f}U_f))] \mid \neg\theta\right] \\
\leq&\gamma\frac{1}{2} + (1-\gamma)1 \\
=&1 - \gamma/2
\end{aligned}
$$

Again, by an averaging argument, there exists an $h_2^*$ such that $\iota_f(h_2^*) \leq 1 - \gamma/2$ □

### 3.2.2 Relating counterfactual and true accuracies

We now show that if the two models agree on both the true and the counterfactual distributions, then they have similar influences.

**Definition 9.** *Given a loss function $l$ and predictors $h$ and $h'$, the expected loss of the $h$ with respect to $h'$, written $err(h, h', X)$, is*

$$err(h, h', X) = \mathop{\mathbb{E}}_{X} \left[ l_{0\text{-}1}(h(X), h'(X)) \right].$$

**Theorem 7.** *If $err(h, h', X) \leq \epsilon_1$, and $err(h, h', X_{-f}U_f) \leq \epsilon_2$, then $|\iota_f(h) - \iota_f(h')| \leq \epsilon_1 + \epsilon_2$*

*Proof.*

$$\begin{aligned}
&|\iota(h, f) - \iota(h', f)| \\
=&\Bigg| \mathop{\mathbb{E}}_{X,U_f} [h(X) \neq h(X_{-f}U_f)] \\
&\qquad\qquad - \mathop{\mathbb{E}}_{X,U_f} [h'(X) \neq h'(X_{-f}U_f)] \Bigg| \\
=&\Bigg| \mathop{\mathbb{E}}_{X,U_f} [|h(X) - h(X_{-f}U_f)|] \\
&\qquad\qquad - \mathop{\mathbb{E}}_{X,U_f} [|h'(X) - h'(X_{-f}U_f)|] \Bigg| \\
&\text{by triangle inequality} \\
\leq& \mathop{\mathbb{E}}_{X,U_f} [|h(X) - h(X_{-f}U_f) - h'(X) + h'(X_{-f}U_f)|] \\
=& \mathop{\mathbb{E}}_{X,U_f} [|h(X) - h'(X) + h'(X_{-f}U_f) - h(X_{-f}U_f)|] \\
&\text{by triangle inequality} \\
\leq& \mathop{\mathbb{E}}_{X,U_f} [|h(X) - h'(X)|] \\
&\qquad\qquad + E_{X,U_f} [|h'(X_{-f}U_f) - h(X_{-f}U_f)|] \\
=& err(h, h', X) + err(h, h', X_{-f}U_f) \\
\leq& \epsilon_1 + \epsilon_2
\end{aligned}$$

□

## 3.3 Counterfactual Active Learning

In this section, we describe an active learning algorithm for training a model that pushes the model towards the desired causal influences. The learning is assisted by two oracles. The first is a feature selection oracle that examines the causal influences of input features, and chooses the feature for which counterfactuals should be labeled. We envision this oracle to be a domain expert that can identify problematic causal influences based on background knowledge of causal factors or ethical norms governing the classification task. The second oracle, similar to a standard active learning oracle, labels counterfactual points with their intended label.

**Algorithm 1** Counterfactual active learning.

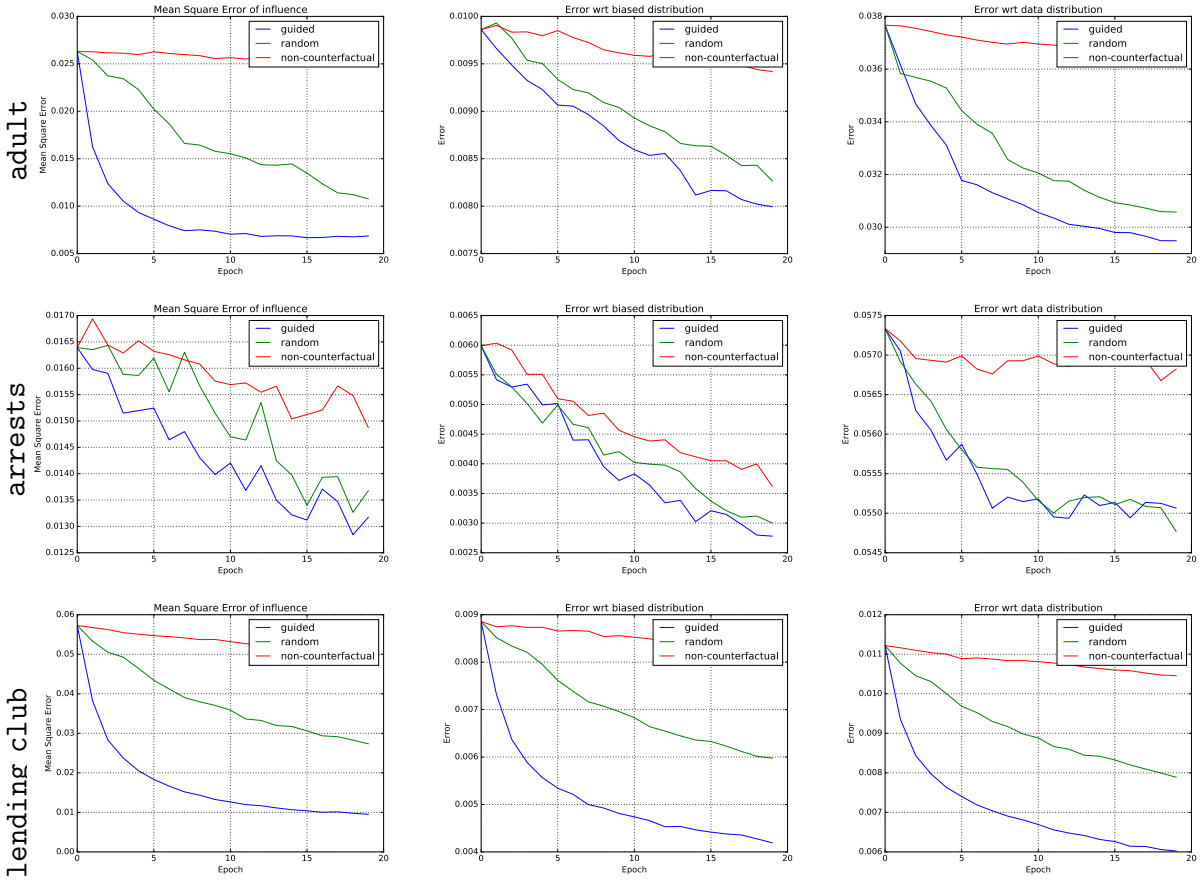
**Require:** Labeling oracle $\mathcal{O}$, Feature selection oracle $\mathcal{F}$
  **procedure** CounterfactualActiveLearning($D, k$)
    $D$: training dataset
    $k$: labeling batch size
    $c \leftarrow \text{train}(D)$
    **repeat**
      $\iota \leftarrow$ feature influences $QII(c, D)$
      $f \leftarrow$ feature $\mathcal{F}(\iota)$
      $\hat{C} \xleftarrow[k]{\$} D_{-f}U_f$
      $\hat{y} \leftarrow$ oracle labels $\mathcal{O}(\hat{C})$
      $D \leftarrow D \cup \langle \hat{C}, \hat{y} \rangle$
      $c \leftarrow \text{train}(D)$
    **until** stopping condition reached
    **return** $c$
  **end procedure**

The active learning process (Algorithm 1), on every iteration, computes the influences of features of a classifier trained on the dataset. The feature selection oracle $\mathcal{F}$ picks a feature. Then $k$ points $\hat{C}$ are picked from the counterfactual distribution, where $k$ is a pre-specified batch size parameter. The parameter $k$ can also be thought of as a learning rate for the algorithm. The $k$ points in $\hat{C}$ are then labeled by the oracle $\mathcal{O}$ and added to the training set. A new classifier is trained on the augmented dataset and this process is repeated until the stopping condition is reached. The stopping condition can either be a pre-specified number of iterations or a convergence condition when the classifier learnt does not show a significant change in influences.

The choice of the feature selection oracle affect the speed of convergence of the algorithm. In our experiments, we consider two feature selection oracles (i) a baseline `random` oracle, that picks features at random for generating counterfactual queries, (ii) a `guided` oracle, that picks the feature that has the highest difference in influence from the true influence. In Section 3.4.2, we demonstrate that an oracle that deterministically picks the feature with the highest difference in influence converges faster than an oracle that picks a feature at random.

The rationale for training the classifier on points from the counterfactual distribution is two-fold. First, by adding points from the counterfactual distribution, the algorithm reduces the divergence between the training distribution and the counterfactual distribution, as a result, constraining the feature influences of the learnt classifier according to Theorem 6. Additionally, by increasing accuracy of the classifier with respect to the labeler on the counterfactual distribution, the influences of the trained classifier are pushed closer to the influence of the labeler (Theorem 7).

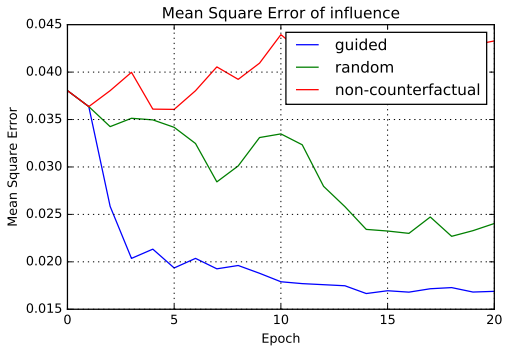
## 3.4 Evaluation



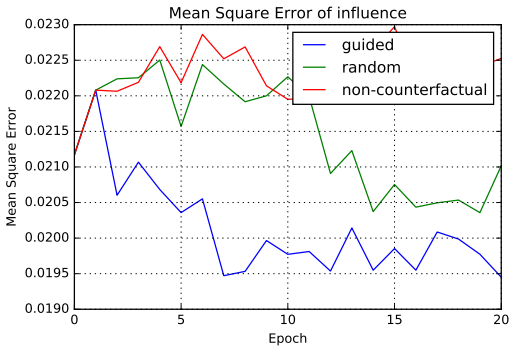

(a) Convergence of difference of influence (b) Convergence of in-distribution error (c) Convergence of out of distribution error

Figure 3.2: Rates of convergence of counterfactual active learning with `guided`, `random`, and `non − counterfactual` settings.

In this section, we evaluate the counterfactual active learning algorithm for linear, decision tree, and random forest models using a synthetic labeler as ground truth. In particular, we demonstrate that after counterfactual active learning, the trained classifier has similar causal influence measures to the labeler. We also show that the classifier can generalize better to out-of-distribution points. This is an important consequence of having causal behavior similar to the labeler. And finally, we demonstrate that the accuracy on the data distribution does not degrade as a result of this additional training.




(a) Convergence of difference of influence




(b) Convergence of difference of influence

Figure 3.3: Rates of convergence of counterfactual active learning with `guided`, `random`, and $\texttt{non}-\texttt{counterfactual}$ settings.

### 3.4.1 Methodology

We evaluate our algorithm by training two predictors. The first predictor provides the ground truth to be used by the oracles. The second predictor is trained on a biased version of the dataset used to train the first predictor. This approach induces a difference in influence in the two predictors. In more detail, the following steps comprise our experimental methodology.

- Given: $D$ a dataset which is a sample from the original distribution,
- Train ground truth model $h_t$ on $D$. This model is used by the labeling oracle to respond to counterfactual queries.
- Select a random predicate $\theta$.
- Construct $D_b$ by excluding points from $D$ that satisfy $\theta$.
- Train predictor $h_b$ on a random subset of $D_b$, leaving the rest for testing and for use by the $\texttt{non}-\texttt{counterfactual}$ baseline described below.
- Perform counterfactual active learning on predictor $h_b$.

The random predicate $\theta$ is chosen by training a short decision tree on the dataset with random labels. $D_b$ is intended to simulate a biased data collection mechanism in order to induce artificial correlations in the dataset. The induced artificial correlations create a gap between the counterfactual distribution and the data distribution, thus making the feature influences unconstrained.

We run the active learning algorithm under the following settings:

- `guided`: At each iteration, the feature selection oracle selects the feature with the highest difference in auQII with respect to the base model.
- `random`: This is a baseline where the feature selection oracle selects a feature at random.
- $\texttt{non}-\texttt{counterfactual}$: This is another baseline where the labeling oracle labels fresh points from $D_b$ as opposed to from the counterfactual distribution.

The experiments presented here are run on the following datasets:

- `adult`: The benchmark adult dataset [85], a subset of census data, is used to predict

income from 13 demographic factors including age and marital status.

- `arrests`: This data set is used to predict a history of arrests using 6 features extracted from the National Longitudinal Survey of Youth [2] such as drug and alcohol use.
- `lending club`: A data set of loans originated by Lending Club [81] is used to predict charge-offs from 19 other financial variables about individuals.

These datasets represent prediction tasks that could be potentially used in settings such as predictive policing or credit, and where data processors are accountable for reasons behind prediction.

All experiments presented here are run with a batch size $t$ of 200 for 20 epochs, and averaged over 50 runs of the algorithm.

### 3.4.2 Results

Figure 3.2 shows the evolution of the active learning algorithm for the three datasets dataset with a logistic regression model. In particular, Figure 3.2a shows the the change of the mean square error of auQII between $h_b$ and $h_t$. This figure shows that the feature influences converge to values close to that of $h_t$ with the `guided` oracle. The `random` oracle also converges but at a slower rate. This result is useful as it indicates that the process does not require the feature selection oracle to pick optimally. The $\texttt{non} - \texttt{counterfactual}$ algorithm does not affect the influence measures significantly. This is to be expected since it retrains using labeled points from within the biased distribution.

In Figures 3.2b and 3.2c, we show the accuracy of the classifier on holdout sets from $D_b$ and $D$ respectively. Figure 3.2b shows that the error on the data distribution does not increase due to this additional training. Further, Figure 3.2c shows that the error on the unbiased dataset $D$ also decreases, even though parts of $D$ are not in the training set. This can be viewed as a side-effect of the model becoming causally closer to the ground truth model. For all three datasets the `guided` oracle leads to faster convergence across the three metrics. The arrests dataset shows similar behavior for the `random` and `guided` oracles which can be attributed to the dataset only containing a small number of features.

Figures 3.3a and 3.3b show the effect of counterfactual active learning on decision trees and random forests. They both show a similar trend to linear models of the causal influences of $h_b$ converging toward those of $h_t$ with counterfactual active learning.

## 3.5 Conclusion

We articulate the problem of covariate shift in causal testing and formally characterize conditions under which it arises. We present an algorithm for counterfactual active learning that addresses this problem. We empirically demonstrate with synthetic labelers that our algorithm trains models that (i) have similar causal influences as the labeler's model, and (ii) generalize better to out-of-distribution points while (iii) retaining their accuracy on in-distribution points.

In this chapter, we assume that the labeling oracle can label all points with equal certainty and cost. However, for points further from the distribution, the labeler might need to perform real experiments in order to label the points. This suggests two interesting directions for future work. The first studies mechanisms for answering counterfactual queries for points far away from the distribution. The second involves designing an algorithm that takes into account the cost of a labeler in the learning process.

# Chapter 4

# Distributional Influence in Continuous Models

In Chapter 2, we proposed a general approach for quantifying the causal influence of inputs in machine learning models, with a particular focus on machine learning classifiers. For machine learning classifiers, we observe that in order to measure causal influences we need to query the model on points outside the distribution it operates on. This requirement results in a covariate shift problem that we address in Chapter 3: approaches to causal testing of classifiers query the model on points where it isn't expected to predict reliably.

However, for continuous and differentiable models including those used for regression, we can observe the causal effects of infinitesimal changes to inputs. For very small values of $\Delta$, we can measure $f(x_1 + \Delta, x_2) - f(x_1, y_2)$ and observe the causal effect of intervening on $x$ by a tiny amount. As $\Delta$ tends to 0, if $(x_1, x_2)$ belongs to the input distribution, then for this infinitesimal intervention, we measure causal influence with zero covariate shift. If this reminds the reader of a partial derivative, then she is not mistaken; this infinitesimal causal measure, when divided by $\Delta$, is exactly the partial derivative of $f$ with respect to $x_1$.

In this chapter, we present *input distributional influence*, an influence measure based on the partial derivative for continuous and differentiable models that captures causality in the model while still retaining distributional faithfulness, i.e. has no covariate shift due to causal testing. This input influence measure is generalized in Chapter 5 to internal factors in models, and evaluated on convolutional neural networks.

**Desiderata.** We begin by describing some intuitive desiderata of measures of influence.

- *Causality:* We intend our influence measures to identify parts of the network, that when changed, have the most effect on outcomes. This suggests a causal view of influence. We focus on the partial derivative on outcomes as a measure of causal influence locally as it measures the change in outcomes corresponding to an infinitesimal change in inputs, while keeping all other inputs fixed.
- *Flexibility:* We wish that with appropriate parameterization, our measures should be useful for answering a general class of causal queries.
- *Distributional Faithfulness:* Models operating on high dimensional spaces such as

neural networks are not expected to behave reliably on instances outside the input distribution. As a result, we wish that our measures only evaluate the behavior of the network on points within the distribution, a property we call *distributional faithfulness*. This property is key to our axiomatic treatment of influence and is reflected in Axioms 2, and 3 in Section 5.4.

Note that both QII and distributional influence share the three desiderata. The third desideratum of distributional faithfulness, however, is discharged differently for the two. For distributional influence, the use of the partial derivative does not evaluate the model on points outside the distribution if the partial derivatives are only measured on points in the distribution. Partial derivatives are not appropriate for QII as they also apply to discontinuous models with discrete inputs, and we need to evaluate models on points outside the distribution. To achieve distributional faithfulness, we train the model on points outside the distribution via counterfactual active learning.

## 4.1 Distributional Input Influence

Distributional input influence is parameterized by a *quantity of interest*, and a *distribution of interest*. The measure is the average partial derivative of the quantity of interest over the distribution of interest.

The distribution and quantity of interest together capture aspects of network behavior that we are interested in explaining. Examples of distributions of interest are: (i) a single instance (influence measure just reduces to the partial derivative at the point) (ii) the distribution of 'cat' images, or (iii) the overall distribution of images. While the first distribution of interest focuses on why a single instance was classified a particular way, the second explains the essence of a class, and the third identifies generally influential inputs over the entire population. A fourth instance is the uniform distribution on the line segment of scaled instances between an instance and a baseline, which yields a measure called Integrated Gradients [129]. Examples of quantities of interest are: outcome towards the 'cat' class (i.e., the network score for the cat class) or comparative outcome towards 'cat' versus 'dog' (i.e., the difference in the network scores for cat and dog classes). The first quantity of interest answers the question of why a particular input was classified as a cat, whereas the second can be helpful in understanding how the network distinguishes 'cat' instances from 'dog' instances.

We represent quantities of interest of networks as continuous and differentiable functions $f$ from $\mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^n$, and $n$ is the number of inputs to $f$. A distributional influence measure, denoted by $\iota_i(f, P)$, measures the influence of an input $i$ for a quantity of interest $f$, and a distribution of interest $P$, where $P$ is a distribution over $\mathcal{X}$.

**Definition 10.** *The distributional input influence $\iota_i(f, P)$ of input $i$ for a quantity of interest $f$, and distribution of interest $P$ is given by*

$$\iota_i(f, P) = \int_{\mathcal{X}} \left.\frac{\partial f}{\partial x_i}\right|_{\mathbf{x}} P(\mathbf{x}) d\mathbf{x}.$$

## 4.2 Axiomatic Characterization

We narrow the space of influence measures using three axioms that characterize desirable properties of influence measures for machine learning models with respect to a quantity and distribution of interest, and then prove that these axioms uniquely define the above measure.

We characterize a measure $\iota_i(f, P)$ that measures the influence of input $i$ for a quantity of interest $f$, and distribution of interest $P$. The first axiom, *linear agreement* states that for linear systems, the coefficient of an input is its influence. Measuring influence in linear models is straightforward since a unit change in an input corresponds to a change in the output given by the coefficient.

**Axiom 1** (Linear Agreement)**.** *For linear models of the form* $f(\mathbf{x}) = \sum_i \alpha_i x_i$, $\iota_i(f, P) = \alpha_i$.

The second axiom, *distributional marginality* states that partial derivatives at points outside the support of the distribution of interest should not affect the influence of an input. This axiom ensures that the influence measure only depends on the behavior of the model on points within the manifold containing the input distribution.

**Axiom 2** (Distributional marginality (DM))**.** *If,* $P(\frac{\partial f_1}{\partial x_i}\Big|_X = \frac{\partial f_2}{\partial x_i}\Big|_X) = 1$, *where* $X$ *is the random variable over instances from* $\mathcal{X}$, *then* $\iota_i(f_1, P) = \iota_i(f_2, P)$.

The third axiom, *distribution linearity* states that the influence measure is linear in the distribution of interest. This ensures that influence measures are properly weighted over the input space, i.e., influence on infrequent regions of the input space receive lesser weight in the influence measure as compared to more frequent regions.

**Axiom 3** (Distribution linearity (DL))**.** *For a family of distributions indexed by some* $a \in \mathcal{A}$, $P(x) = \int_{\mathcal{A}} g(a) P_a(x) da$, *then* $\iota_i(f, P) = \int_{\mathcal{A}} g(a) \iota_i(f, P_a) da$.

We can show that the only influence measure that satisfies these three axioms is the weighted partial derivative of the input probability distribution.

**Theorem 8.** *The only measure that satisfies linear agreement, distributional marginality and distribution linearity is given by*

$$\iota_i(f, P) = \int_{\mathcal{X}} \frac{\partial f}{\partial x_i}\Big|_{\mathbf{x}} P(\mathbf{x}) d\mathbf{x}.$$

*Proof.* Choose any function $f$ and $P_{\mathbf{a}}(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{a})$, where $\delta$ is the Dirac delta function on $\mathcal{X}$. Now, choose $f'(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}_i}|_{\mathbf{a}} x_i$. By linearity agreement, it must be the case that, $\iota(f', P_{\mathbf{a}}(\mathbf{x})) = \frac{\partial f}{\partial x_i}|_{\mathbf{a}}$. By distributional marginality, we therefore have that $\iota_i(f, P_{\mathbf{a}}) = \iota_i(f', P_{\mathbf{a}}) = \frac{\partial f}{\partial x_i}|_a$. Any distribution $P$ can be written as $P(\mathbf{x}) = \int_{\mathcal{X}} P(\mathbf{a}) P_{\mathbf{a}}(\mathbf{x}) d\mathbf{a}$. Therefore, by the distribution linearity axiom, we have that $\iota(f, P) = \int_X P(\mathbf{a}) \iota(f, P_a) da = \int_{\mathcal{X}} P(\mathbf{a}) \frac{\partial f}{\partial x_i}|_{\mathbf{a}} d\mathbf{a}$. □

## 4.3 Related Work

As mentioned in Chapter 2, influence measures are studied in cooperative game theory as solutions to the problem of attribution of outcomes to participants and has applications to

a wide range of settings including revenue division and voting. In Chapter 2, we viewed features as participating in finite cooperative games where valuation functions $v(S)$ capture the outcomes generated when agents in set $S$ participate; a feature belongs in $S$ if it is not intervened on. In this chapter focusing on continuous models, we draw ideas from the theory of infinite cooperative games, where valuation functions are of the form $v(\mathbf{x})$, and $x_i$ represents the numerical contribution of agent $i$.

We highlight ideas drawn from this body of work and differences in terms of two key properties of influence measures: the *marginality principle*, and *efficiency*.

The *marginality principle* [146] states that an agent's attribution only depends on its own contribution to the output. Formally, this is stated as: if the partial derivatives with respect to an agent of two functions are identical throughout, then they have identical attributions for agent $i$. Our axiom of distributional marginality (DM) is a weaker form of this requirement that only requires equality of attribution when partial derivatives are same in the distribution.

A second property, called *efficiency*, which is especially important for revenue division, is that attributions add up to the total value generated. This ensures that no value is left unattributed. The marginality principle, along with efficiency uniquely define the *Aumann-Shapley Value*[9]. In [129], the Aumann-Shapley Value is used for attributions with efficiency as a justification. While it is unclear that efficiency is an essential requirement in our setting, the Aumann-Shapley value can be recovered in our framework by choosing the distribution of interest as the uniform distribution on the line segment joining an instance $\mathbf{x}$ and a baseline image $\mathbf{b}$. Certain choices of baselines can be problematic from the point of view of distributional faithfulness, since the line segment of linear combinations between them might lie significantly out of distribution. The particular baseline chosen in [129] is the zero vector, where the line segment represents scaled images, and could be reasonably called within distribution.

Partial derivatives (gradients) have also been used as a measure of influence in prior work for interpreting machine learning models. In particular, for convolutional neural networks, a common approach to interpreting their functioning is to visualize gradients [10, 122, 129], or some combination of gradients with internal values [10, 127, 148]. A more comprehensive comparison with this body of work is deferred to Chapter 5.

# Part II

# Indirect Use

# Chapter 5

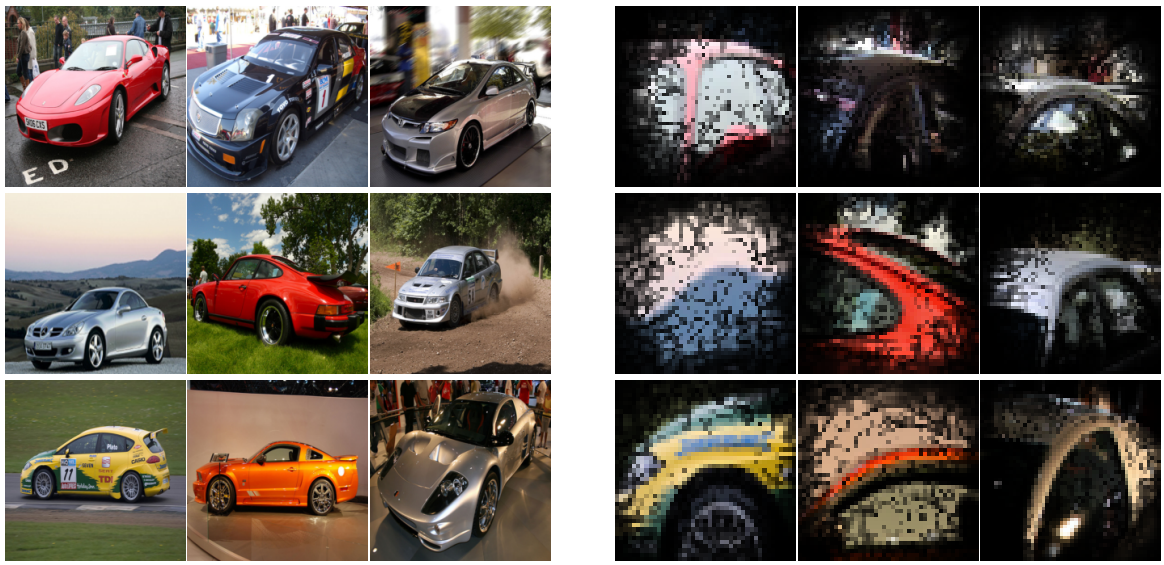
# Explanations for CNNs

We study the problem of explaining a class of behavioral properties of deep neural networks, with a focus on convolutional neural networks operating on images. This problem has received significant attention in recent years with the rise of deep networks and associated concerns about their opacity [75]. Explanations that provide insight into the reasons behind incorrect network behavior play an important role in mitigating this opacity.

A growing body of work on explaining deep convolutional network behavior is based on mapping models' prediction outputs back to relevant regions in an input image. This is accomplished in various ways, such as by visualizing gradients [10, 122, 129] or backpropagation [10, 127, 148]. Recently [111] proposed fitting simpler interpretable models around a test point to predict relevant input regions. An appealing feature of these approaches is that they capture input influence. However, because these approaches relate instance-specific features to instance-specific predictions, the explanations that they produce do not generalize beyond a single test point (see Section 5.2, Figure 5.2).

An orthogonal approach is to visualize the features learned by networks by identifying input instances that maximally activate an internal neuron, by either optimizing the activation in the input space [89, 96, 122], or searching for instances in a dataset [55]. Importantly, this type of explanation gives insight into the higher-level concepts learnt by the network, and naturally generalizes across instances and classes. However, this approach does not relate these higher-level concepts to predictions that they cause. Indeed examining activations alone is not sufficient to do so (see Section 5.2, Table 5.1).

Continuing with our template of combining influence and interpretation, we develop *influence-directed explanations* for deep networks to combine the positive attributes of these two lines of work. Our approach peers inside the network to identify neurons with high *influence* on the model's behavior, and then uses existing techniques (e.g., visualization) to provide an *interpretation* for the concepts they represent. We generalize the distributional input influence measure presented in Chapter 4 to internal neurons, and present a *distributional influence* measure that allows us to identify which neurons are most influential in determining the model's behavior on a given distribution of instances. From this we are able to identify the learned concepts that cause the network to behave characteristically, for example, on the distribution of instances that share a particular label.



**(a)** **(b)**

Figure 5.1: Images of cars labeled "sports car" by VGG16 ImageNet model (a) and receptive fields of the most influential feature map on a comparative quantity that characterizes the model's tendency to predict "sports car" over "convertible" (b). In most cases, the features within the field contain the top of the car, which is the key distinctive concept between these classes.

Figure 5.1 demonstrates on a VGG16 [121] model trained on the ImageNet dataset [114] the capability of influence-directed explanations to extract meaningful insight about the network's inner workings. We measure the influence of feature maps at the `conv4_1` layer on the network's tendency to predict "sports car" over "convertible". The images in Figure 5.1(b) are computed by rendering the receptive field of the *most* influential map in the original feature space for the corresponding image in 5.1(a). The results coincide with an intuitive understanding of the distinction between these classes: the depicted interpretation highlights the portion of the image depicting the car's top.

Our empirical evaluation demonstrates that influence-directed explanations (i) extract influential concepts that generalize across instances whereas input-influence-based explanations fail to do so (Section 5.2.1); (ii) reveal the essence of how the network views a class (Section 5.2.2); (iii) isolate features that the network uses to make predictions and distinguish related instances (Section 5.3.1, 5.3.2) and (iv) assist in understanding misclassifications (Section 5.3.3). In each case, our influence-directed explanations leverage the ability to measure internal influence to produce useful explanations that would not have been possible otherwise.

These influence-directed explanations can be approximated with a constant number of gradient calls, where the constant depends on the number of parameters, and the accuracy required. Gradient calls are relatively inexpensive for even large networks through optimized parallel implementations on GPUs.

## 5.1 Influence

In this section, we generalize the distributional input influence measure presented in Chapter 4, and we propose *distributional influence*. In addition to a *quantity of interest*, and a *distribution of interest*, Distributional influence is parameterized by a *slice* of the network (e.g. a particular layer). The measure is the average partial derivative of the quantity of interest over the distribution of interest at the slice. We describe the measure and its parameters below. In Section 5.4, we justify this family of measures by proving that these are the only measures that satisfy some natural properties.

The slice parameter exposes the internals of a network, and allows one to compute influence with respect to intermediate neurons, a significant departure from prior work. Importantly, as opposed to input pixels, as internal neurons can represent higher-level concepts, influential internal neurons allow explanations to be more general rather than being specific to single instances. In Section 5.5, we compare this measure to others used in prior work.

Recall that the distribution and quantity of interest together capture aspects of network behavior that we are interested in explaining. Examples of distributions of interest are: (i) a single instance (influence measure just reduces to the gradient at the point) (ii) the distribution of 'cat' images, or (iii) the overall distribution of images. While the first distribution of interest focuses on why a single instance was classified a particular way, the second explains the essence of a class, and the third identifies generally influential neurons over the entire population. A fourth instance is the uniform distribution on the line segment of scaled instances between an instance and a baseline, which yields a measure called Integrated Gradients [129]. Examples of quantities of interest are: outcome towards the 'cat' class (i.e., the network score for the cat class) or comparative outcome towards 'cat' versus 'dog' (i.e., the difference in the network scores for cat and dog classes). The first quantity of interest answers the question of why a particular input was classified as a cat, whereas the second can be helpful in understanding how the network distinguishes 'cat' instances from 'dog' instances.

Similar to Chapter 4, we represent quantities of interest of networks as continuous and differentiable functions $f$ from $\mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^n$, and $n$ is the number of inputs to $f$. A distributional influence measure, denoted by $\iota_i(f, P)$, measures the influence of an input $i$ for a quantity of interest $f$, and a distribution of interest $P$, where $P$ is a distribution over $\mathcal{X}$. Next, we define a slice of a network. A particular layer in the network can be viewed as a slice. More generally, a slice is any partitioning of the network into two parts that exposes its internals. Formally, a slice $s$ of a network $f$ is a tuple of functions $\langle g, h \rangle$, such that $h : \mathcal{X} \to \mathcal{Z}$, and $g : \mathcal{Z} \to \mathbb{R}$, and $f = g \circ h$. The internal representation for an instance $\mathbf{x}$ is given by $\mathbf{z} = h(\mathbf{x})$. In our setting, elements of $\mathbf{z}$ can be viewed as the activations of neurons at a particular layer.

**Definition 11.** *The influence of an element $j$ in the internal representation defined by $s = \langle g, h \rangle$ is given by*

$$\iota_j^s(f, P) = \int_{\mathcal{X}} \left. \frac{\partial g}{\partial z_j} \right|_{h(\mathbf{x})} P(\mathbf{x}) d\mathbf{x} \tag{5.1}$$

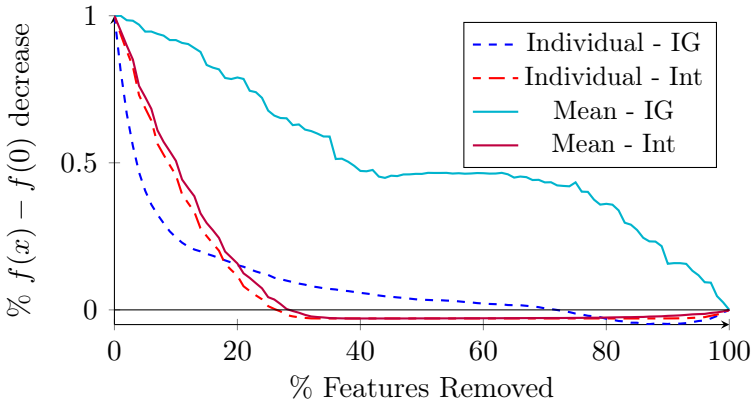



Figure 5.2: Plot of the decrease in the function value (before the softmax), $f(x)$, as features are removed in order of influence. The model is based on LeNet [78] trained on the MNIST dataset. In each case, the most influential feature or hidden unit is incrementally removed, and the resulting value of $f(x)$ is depicted on the vertical axis. The vertical axis was normalized so that the average value of $f(x)$ on class "3" is 1, and 0 is the value of $f(0)$. The dashed curves depict the quantity when influence is measured for each instance individually (averaged across all instances of class "3"), and the solid curves when influence is with the empirical distribution of the entire class as the distribution of interest. Plots are shown for both integrated gradients (**IG**), as well as for internal influence (**Int**) for a slice at the first fully-connected layer of the network.

## 5.2 Identifying Influential Concepts

The influence measure defined in Section 5.1 is parameterized by a distribution of interest $P$ (Equation 5.1) over which the measure is taken. By selecting $P$ to be a point mass, the resulting measurements characterize the importance of features or components for the model's behavior on a single instance. Any meaningful interpretation of these measurements can refer only to that instance, and thus reflect specific features and concepts that may not generalize across a class. Defining the distribution of interest with support over a larger set of instances will yield explanations that capture the factors common to network behaviors across the corresponding population of instances. These explanations capture the "essence" of what the network learned about that population, and can be used to identify the concepts that are most relevant to the network's behavior on it.

### 5.2.1 Effectiveness of Internal Influence

Figure 5.2 quantifies the degree to which internal units identified using influence correspond to relevant general concepts, compared against the influence measurements obtained using integrated gradients [129]. The curves are computed by "turning off" either input features (IG) or units in the network in the order determined by their influence. We adapted this approach from [116] for hidden units by intervening to set their activation to 0. The vertical

axis depicts the percentage dropoff of the network's output prior to softmax, against the percentage of features that have been removed by order of their influence.

We evaluated this measure on instances of the number 3 from MNIST on a LeNet model [78]. We selected integrated gradients as our point of comparison because we found that it outperformed comparable instances discussed in the related work. We calculated influence in two ways to characterize the difference between instance-specific and general measurements. In the cases labeled "Individual", we measure influence for each instance individually and rank features and units accordingly. For those labeled "Mean" we measure influence using a distribution of interest corresponding to the class in question, and rank features and units for each instance according to the resulting ordering.

Comparing the individual and mean results tells us how well the explanations generalize across the class. If the individual explanations significantly outperform their respective mean explanations, then we may conclude that the distributional influences failed to identify concepts that are relevant to many instances in the class.

The results in Figure 5.2 show a very small gap between the performance of the instance-specific and mean cases. This suggests that units deemed relevant to the class on-average also tend to matter consistently across instances in that class. Compared with integrated gradients [129], which operates on the input features, the class-wide average influences computed from internal nodes outperform significantly. This is not particularly surprising, but the size of the gap illustrates the additional explanatory power of influential internal units.

Figure 5.2 also shows that using integrated gradients results in a more rapid initial dropoff. Regardless, the area under the curve for the respective internal influence is approximately 8% lower than that of integrated gradients, and the function value drops to its baseline value more quickly by a factor of 4. We observed that influence is distributed fairly evenly across fewer units in the layer in question for these experiments, whereas in the bottom input layer there are a relatively small number of features that account for most of the influence, and a longer tail of features that have a small amount of influence. Finally, the sharp dropoff of IG may be additionally explained by the observation that on many MNIST instances it may be possible to change classification by changing relatively few pixels [103], whereas this phenomenon may not be as pronounced on other datasets.

### 5.2.2 Validating the "Essence" of a Class

As is apparent in Figure 5.6, it is often the case that relatively few units are highly influential towards a particular class. In such cases, we refer to this as the "essence" of the class, as the network's behavior on these classes can be understood by focusing on these units. To validate this claim, we demonstrate that these units can be isolated from the rest of the model to extract a classifier that is more proficient at distinguishing class instances from the rest of the data distribution than the original model. To this end, we introduce a technique for compressing models using influence measurements to yield class-specific "expert" models that demonstrate the essence of that class learned by the model.

Given a model, $f$, with softmax output, and slice, $\langle g, h \rangle$, where $g : \mathcal{Z} \to \mathcal{Y}$, let $M_h \in \mathcal{Z}$ be a 0-1 vector. Intuitively, $M_h$ masks the set of units at layer $h$ that we wish to retain, and

| | 5 | 10 | 20 | 40 | 80 | 120 | 160 | 240 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.00 | 0.39 | 0.78 | 0.82 | 0.39 | 0.08 | 0.06 | 0.00 |
| 10 | 0.00 | 0.33 | 0.77 | 0.83 | 0.46 | 0.18 | 0.11 | 0.04 |
| 20 | 0.00 | 0.33 | 0.73 | 0.84 | 0.53 | 0.20 | 0.18 | 0.04 |
| 40 | 0.00 | 0.23 | 0.69 | 0.86 | 0.63 | 0.40 | 0.23 | 0.04 |
| 80 | 0.00 | 0.21 | 0.60 | 0.83 | 0.76 | 0.58 | 0.45 | 0.18 |
| 120 | 0.00 | 0.15 | 0.52 | 0.80 | 0.83 | 0.68 | 0.53 | 0.33 |
| 160 | 0.00 | 0.08 | 0.43 | 0.77 | 0.87 | 0.81 | 0.66 | 0.50 |
| 240 | 0.00 | 0.02 | 0.28 | 0.67 | 0.85 | 0.89 | 0.86 | 0.74 |
| 320 | 0.00 | 0.00 | 0.23 | 0.53 | 0.81 | 0.88 | 0.92 | 0.84 |
| 640 | 0.00 | 0.00 | 0.00 | 0.32 | 0.58 | 0.73 | 0.83 | 0.88 |

**(a)**

Figure 5.3: $F_1$ score for experts on a randomly-selected class from ImageNet using the VGG16 network. The rows and columns correspond to $\alpha$ and $\beta$ respectively. The results indicate that while selecting larger sets of units for compression does lead to increased performance, the returns diminish rapidly and good experts can be identified using a small set of units.

so is 1 at all locations corresponding to such units and 0 everywhere else. Then the *slice compression* $f_{M_h}(X) = g(h(X) * M_h)$ corresponds to the original model after discarding all units at $h$ not selected by $M_h$. Given a model $f$, we obtain a binary classifier $f^i$ for class $L_i$ (corresponding to softmax output $i$) by projecting the softmax output at $i$, in addition to the sum of all other outputs: $f^i = (f|_i, \sum_{j \neq i} f|_j)$, where $f|_i$ is the projection of the model's softmax output to its $i^{th}$ coordinate.

**Class-specific Experts** For the sake of this discussion, we define a class-wise *expert* for $L_i$ to be a slice compression $f_{M_h}$ whose corresponding binary classifier $f^i_{M_h}$ achieves better recall on $L_i$ than the binary classifier $f^i$ obtained by $f$, while also achieving approximately the same recall. We demonstrate that the influence measurements taken at slice $\langle g, h \rangle$ over a distribution of interest, $P_i$, conditioned on class $L_i$ yields an efficient heuristic for extracting experts from large networks.

In particular, we compute $M_h$ by measuring the slice influence (Equation 5.1) over $P_i$ using the quantity of interest $g|_i$. Given parameters $\alpha$ and $\beta$, we select $\alpha$ units at layer $h$ with the largest positive influence, and $\beta$ units with the greatest negative influence (i.e., greatest magnitude among those with negative influence). $M_h$ is then defined to be zero at all positions except those corresponding to these $\alpha + \beta$ units. In our experiments, we obtain concrete values for $\alpha$ and $\beta$ by a parameter sweep, ultimately selecting those values that yield the best experts by the criteria defined above. Figure 5.3 shows the F1 score obtained on a randomly-selected class as a function of $\alpha$ and $\beta$. Notably, both measures plateau with relatively few units in the compressed model, indicating that good parameters can be identified quickly in practice by selecting a small portion of the units at the chosen layer.

| Class | Orig. | Act. | Infl. |
|---|---|---|---|
| Chainsaw (491) | .14 | 0. | .71 |
| Bonnet (452) | .62 | 0. | .92 |
| Park Bench (703) | .52 | 0. | .71 |
| Sloth Bear (297) | .36 | 0. | .75 |
| Pelican (144) | .65 | 0. | .95 |

Table 5.1: Model compression recall for five randomly-selected ImageNet classes. Columns marked Orig. correspond to the original model, Act. to experts computed using activation levels, and Infl. to experts computed using influence measures. Precision in all cases was 1.0.

Table 5.1 shows the recall of experts found in this way for five randomly selected ImageNet classes, as well as the recall of the original model on each class and on experts computed using activations, rather than influence. Precision is not shown because in all cases it was 1.0. This shows that the top and bottom influential neurons are sufficient to capture the concepts embodied in a particular layer that discriminate a given class from the others. Removing non-influential neurons yields significantly higher recall than the baseline model, and moreover *activation levels are not a meaningful and consistent indication of the relevance of a neuron.* In other words, measuring internal influence is an effective way to identify the concepts that the network learned in order to discriminate classes from each other.

### 5.2.3 Disappearing Experts

The results discussed so far demonstrate that internal distributional influence measurements can be used to identify relevant concepts that generalize across instances, and distinguish between classes. In this section we show that the concepts identified in this way often represent input-space features that are interpretable by domain experts as important for correctly classifying instances, and can be identified as such even when it is not possible to interpret those concepts reliably in pixel space.

Building on the work of [58, 109], we trained an Inception network [130] to diagnose the severity of diabetic retinopathy in color retinal fundus images [69]. Diabetic Retinopathy (DR) is a medical condition characterized by damage to the retina occurring due to diabetes. DR is classified on a scale from 1 to 5 [6], with class 1 corresponding to the absence of symptoms and class 5 being the most severe presentation. In the Kaggle dataset used to train our model, class-1 is the most common (39,539 instances) and the remaining classes (14,042 instances) distributed relatively evenly. Class-2, the least severe positive diagnosis, is characterized by the presence of visible microaneurysms only, with no other symptoms present on the fundus image [6] that distinguish it from class-1. Due to their small size in

the pixel space, validating that they have been identified by an influence measurement is challenging because it is not possible to visualize them well.

To address this challenge, we created a dataset to control for the presence of microaneurysm features that characterize class-2 instances. Specifically, we preprocessed all images with a minor Gaussian blur (radius 2) to remove the corresponding visual features, and trained a second model on the dataset generated by this intervention.

The model trained on the original dataset behaved as expected, and achieved a non-trivial recall of class-2 instances (approximately 15%). We were also able to extract an expert for each class from this model that improved on the recall of the original model, demonstrating that internal influence measures identified distinctive concepts in this case.

The model trained on the intervened dataset displayed classification behavior consistent with our expectation that applying small-radius Gaussian blur removes microaneurysm features. Namely, the intervened model classified none of the instances in the validation set as class-2, and instead classified 95% of the true class-2 instances as class-1. Moreover, when we applied the same strategy for extracting an expert for class-2, we were unable to find a set of influential neurons that achieved better (i.e., non-zero) recall. This was the case even if we relax the criteria for selecting experts to allow for reduced precision.

To summarize, by controlling for the presence of an important concept in the source data, it is often possible to characterize the concept represented by internal units by testing for "disappearing experts" in retrained models.

## 5.3 Explaining Instances

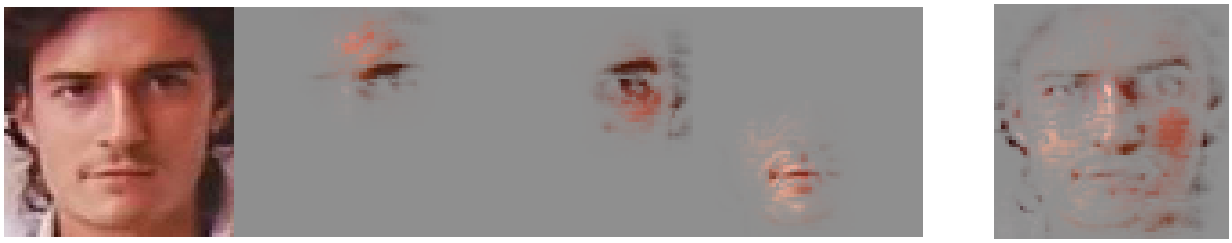
In this section we demonstrate that the learned concepts identified by measuring influence on internal units are useful when explaining model behavior on individual instances.

### 5.3.1 Focused Explanations from Slices

As discussed in Section 5.2, computing the influence on a slice of the network (Equation 5.1) lets us determine how relevant neurons in intermediate layers are to a particular network behavior. In particular, given an image and the network's prediction on that image, the influence measurements for a slice can reveal which features or concepts present in that image were relevant to the prediction.

Figure 5.4(a) shows the results of interpreting the influences taken on a slice of the VGG16 [121] corresponding to an intermediate convolutional layer (`conv4_1`). In this example we take the three most influential units for the quantity of interest corresponding to the correct classification of this image. More precisely, the quantity of interest used in this example corresponds to $f|_L$, i.e., the projection of the model's softmax output to the coordinate corresponding to the correct label $L$ of this instance. The visualization for each of these units was then obtained by measuring the influence of the input pixels on these units along each color channel, and scaling the pixels in the original image accordingly.

Because convolutional units have a limited receptive field, the resulting interpretation shows distinct regions in the original image, in this case corresponding to the left eye and



**(a)** **(b)**

Figure 5.4: **(a)** Interpretation of the three most influential units from a slice corresponding to a convolutional layer (`conv4_1`), for the VGG16 [121] network. **(b)** Explanation based on integrated gradients [129], taken on the same network and image. The interpretation in both cases was computed by scaling pixels in the original image using the results of either method.

**(a)** **(b)**

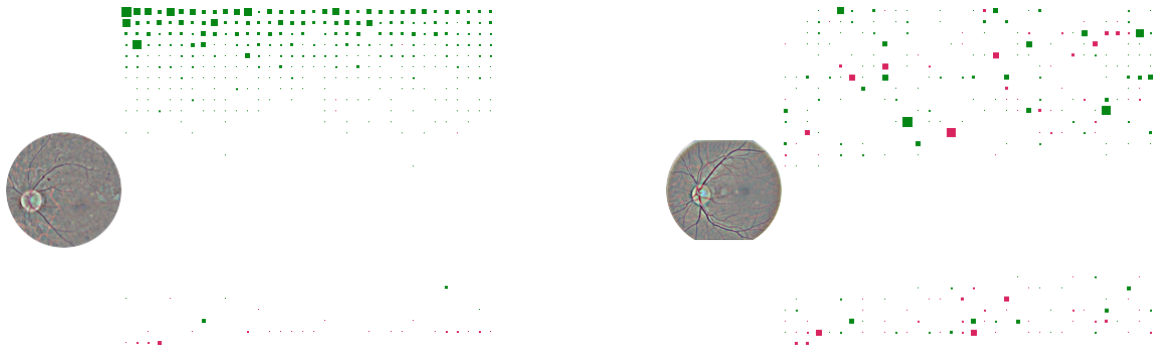
Figure 5.5: **(a)** Comparative explanation of classes "sports car" and "convertible" taken from the top-three most influential units at the `conv4_1` layer (VGG16 [121]). **(b)** Explanation computed using the quantity of interest corresponding to "sports car" on the same instance used in **(a)**.

mouth, that were most relevant to the model's predicted classification. When compared to the explanation provided by integrated gradients [129] on input features shown in Figure 5.4(b), it is evident that the explanation based on network's internal units is better at localizing the features used by the network in its prediction.

### 5.3.2 Comparative Explanations

Influence-directed explanations are parameterized by a quantity of interest, corresponding to the function $f$ in Equation 5.1. Changing the quantity of interest gives additional flexibility in the characteristic explained by the influence measurements and interpretation. One class of quantities that is particularly useful in answering counterfactual questions such as, "Why was this instance classified as $L_1$ rather than $L_2$?", is given by the *comparative quantity*.

More precisely, if $f$ is a softmax classification model that predicts classes $L_1, \ldots, L_n$, then the comparative quantity of interest between classes $L_i$ and $L_j$ is $f|_i - f|_j$. When used in Equation 5.1, this quantity captures the tendency of the model to classify instances as $L_i$ over $L_j$. In Section 5.2 we used this quantity to show that internal influences computed over



(a) (b)

Figure 5.6: Distributional influence measurements taken on DR model (Section 5.2.3) at bottom-most fully connected layer. To compute the grid, the distribution of influence was conditioned on class 5 (a) and class 1 (b). Figure (a) depicts an instance from class 5 that was correctly classified as such, and (b) an instance from class 5 that was incorrectly classified as class 1. In (a) the influences depicted in the grid align closely with the class-wide ordering of influences, whereas in (b) they are visibly more random. White space in the middle of the grid corresponds to units with no influence on the quantity.

a distribution capture high-level concepts learned by the network, and here we demonstrate that they are also useful when explaining individual predictions.

Continuing the example discussed in Section 5.2, Figure 5.5(a) shows an example of a comparative explanation taken for a VGG16 [121] model trained on the ImageNet dataset [114]. The original instance shown on the left is labeled in the "sports car" leaf node of the ImageNet hierarchy. We measured influence using a comparative quantity against the leaf class "convertible", using a slice at the `conv4_1` convolutional layer. The interpretation was computed on the top-three most influential units at this layer in the same way as discussed in Section 5.3.1.

As in the examples from Figure 5.1, the receptive field of the most influential unit corresponds to the region containing the hard top of the vehicle, which is understood to be its most distinctive feature according to this comparative quantity. Figure 5.5(b) shows an interpretation for the same instance computed using influence measurements taken from the quantity of interest $f|_L$ (i.e., the same quantity used in Section 5.3 and implicitly used by integrated gradients [129]). While both explanations capture features common to cars, only the comparative explanation isolates the distinctive elements of the feature space.

### 5.3.3 Understanding Misclassification

Influential distributional concepts can also lead to insights about misclassification behavior on particular instances. Figure 5.6 depicts an example of such an explanation. These visualizations were generated by measuring influence on a slice corresponding to the bottom-most fully-connected layer of a Diabetic Retinopathy (DR) model (see Section 5.2.3 for details on this model).

The quantity of interest $f|_i$ corresponds to a particular class outcome, and the distribution was conditioned on the corresponding class label. The units in that layer were then sorted according to their influence on the conditioned distribution, with the top-left corner corresponding to the largest positive influence, and the bottom-right to the largest negative influence. For a specific instance (shown on the left of Figures 5.6(a) and (b)), influences at that layer were then measured, and the magnitude and sign of the corresponding unit in the class-wide ordering is depicted by the size and shape of the box at that position: large boxes denote larger magnitude, whereas green boxes denote positive sign and red negative.

Figure 5.6(a) depicts this for an instance of class 5 that was correctly classified, whereas 5.6(b) for an instance of the same class that was incorrectly classified. The differences are readily apparent: influences for the correctly-classified instance align closely with the distributional order, whereas they do not in the incorrectly-classified case. This suggests a notion of an instance being classified for a "correct reason," as in 5.6(a), or an "incorrect" or "anomalous" reason, as in 5.6(b), which can be determined by identifying influential high-level concepts.

## 5.4 Axiomatic Justification of Measures

In this section we justify the family of measures presented in 5.1, by defining a set of natural properties that an influence measure should satisfy, and proving a tight characterization of these measures. We generalize the distributional input influence measure presented in Chapter 4 to general slices, and address the case where influence is measured with respect to internal neurons. We again take an axiomatic approach, with two natural invariance properties on the structure of the network.

The first axiom states that the influence measure is agnostic to how a network is sliced, as long as the neuron with respect to which influence is measured is unchanged. Below, the notation $\mathbf{x}_{-i}$ refers to the vector $\mathbf{x}$ with element $i$ removed.

Two slices $s_1 = \langle g_1, h_1 \rangle$ and $s_2 = \langle g_2, h_2 \rangle$ are $j$-equivalent if for all $\mathbf{x} \in \mathcal{X}$, and $z_j \in \mathcal{Z}_j$, $h_1(\mathbf{x})_j = h_2(\mathbf{x})_j$, and $g_1(h_1(\mathbf{x})_{-j} z_j) = g_2(h_2(\mathbf{x})_{-j} z_j)$. Informally, two slices are $j$-equivalent as long as they have the same function for representing $z_j$, and the causal dependence of the outcome on $z$ is identical.

**Axiom 4** (Slice Invariance)**.** *For all $j$-equivalent slices $s_1$ and $s_2$, $\iota_j^{s_1}(f, P) = \iota_j^{s_2}(f, P)$.*

The second axiom equates the distributional input influence of an input with the internal influence of a perfect predictor of that input. Essentially, this encodes a consistency requirement between inputs and internal neurons that if an internal neuron has exactly the same behavior as an input, then the internal neuron should have the same influence as the input.

**Axiom 5** (Preprocessing)**.** *Consider $h_i$ such that $P(X_i = h_i(X_{-i})) = 1$. Let $s = \langle f, h \rangle$, be such that $h(x_{-i}) = x_{-i} h_i(x_{-i})$, which is a slice of $f'(\mathbf{x}_{-i}) = f(\mathbf{x}_{-i} h_i(\mathbf{x}_{-i}))$, then $\iota_i(f, P) = \iota_i^s(f', P)$.*

We can now show that the only measure that satisfies these two properties is the one presented above.

**Theorem 9.** *The only measure that satisfies slice invariance and preprocessing is*

$$\iota_j^s(f, P) = \int_{\mathcal{X}} \left.\frac{\partial g}{\partial z_j}\right|_{h(\mathbf{x})} P(\mathbf{x}) d\mathbf{x}$$

## 5.5 Related Work

Prior work on interpreting CNNs has focused on answering two questions: (i) given an input image, what part of the instance is relevant to a particular neuron? and (ii) what maximizes the activation of a particular neuron?

**Identifying influential regions** One approach to interpreting predictions for convolutional networks is to map activations of neurons back to regions in the input image that are the most relevant to the outcomes of the neurons. Possible approaches for localizing relevance are to (i) visualize gradients [10, 122, 129] (ii) propagate activations back using gradients [10, 127, 148], (iii) learning interpretable models predicting the effect of the presence of regions in an image [111]. Because these approaches relate instance-specific features to instance-specific predictions, their results do not generalize beyond a single test point.

**Visualization by maximizing activation** An orthogonal approach is to visualize features learnt by networks by identifying input instances that maximally activate a neuron, achieved by either optimizing the activation in the input space [89, 96, 122], or searching for instances in a dataset [55]. Examining causal influence of neurons rather than their activations better identifies neurons used by a network for classification. Experiments in Section 5.2.2 demonstrate why examining activations fails to identify important neurons.

Table 5.2 presents a detailed comparison of the influence-directed explanation framework presented here to related prior work measuring influence for CNNs. The leftmost three columns describe framework properties for which some measures allow flexibility. First, our explanations are parametric in a distribution of interest, allowing us to explain the network behavior at different levels of granularity (e.g., an instance or a particular class). Second, our explanations are parametric in a quantity of interest that allow us to provide explanations for different behaviors of a system, as opposed to instance outcomes. Cells marked $\checkmark^*$ in these columns denote that these measures offer a limited form of flexibility along this dimension with the choice of an appropriate baseline. Third, examining the influence of internal neurons plays an important role here because they capture more general concepts, and we demonstrate it is possible to identify "expert" neurons for certain distributions (Section 5.2.2). In contrast, the frameworks proposed for Integrated Gradients [129], Sensitivity Analysis [122], and the Simple Taylor Decomposition [10] are based on attributing relevance solely to the input features. For deconvolution [148] and guided backpropogation [127], these cells are marked a $\checkmark^\dagger$ as internal influences are used only as an intermediary to computing input influence.

| | *Explanation framework properties* | | | *Influence properties* | |
|---|---|---|---|---|---|
| | **Quantity** | **Distribution** | **Internal** | **Faithfulness** | **Sensitivity** |
| influence-directed | ✓ | ✓ | ✓ | ✓* | ✓ |
| integrated gradients | | ✓* | | ✓* | ✓ |
| simple Taylor | | ✓* | | ✓* | ✓ |
| sensitivity analysis | | | | ✓ | |
| deconvolution | | | ✓† | ✓ | |
| guided backpropagation | | | ✓† | ✓ | |
| relevance propagation | | | ✓† | ✓ | ✓* |

Table 5.2: Comparison of the influence-directed explanations proposed here to prior related work including Integrated Gradients [129], Sensitivity Analysis [122], Deconvolution [148], Layer-wise Relevance Propagation [10], and Simple Taylor Decomposition [10]. The first three columns refer to capabilities of the corresponding explanation framework: the flexibility in the choice of **Quantity** of interest and **Distribution** of interest over instance, and the ability to examine the role of **Internal** neurons. The latter two columns describe properties of the influence measure used to build explanations: **Faithfulness** refers to distributional faithfulness; **Sensitivity** requires that if two instances differ in one feature and yield different predictions, then that feature is assigned non-zero influence. $*$ denotes that the framework may have the feature under certain parameterizations, and † denotes that the framework measures internal influence only as an intermediary step to computing feature influence.

The rightmost two columns in Table 5.2 reflect that our choice of influence measures are guided by axiomatic choices different from those in prior work [129]. A notable difference stems from our distributional faithfulness criteria, which imposes a weaker distribution marginality principle than the marginality principle imposed by Integrated Gradients. A practical consequence of this criteria for Integrated Gradients is that only certain choices of baselines satisfy faithfulness (denoted by $\checkmark^*$). Notably, several of the frameworks make use of influence measures that do not satisfy sensitivity under any circumstances; this matter is described in further detail in [129]. Measures that do not satisfy sensitivity can be problematic in practice because they may fail to identify features or components that are causally-relevant to the quantity of interest, leading to "blind spots" or a focus on irrelevant features.

# Chapter 6

# Proxy Use

While the first part of this work focuses on the use of input factors which are explicitly provided, information can be used indirectly through proxies. Proxies allow a data processor to effectively infer protected information types and use them even when they are not explicitly provided.

We use an example, inspired by the Target case [41], to motivate the challenges in defining proxy use in automated decision-making systems. Consider a pharmacy within a retail store, such as Target. We consider various ways in which the retail store and its pharmacy may use information about the pregnancy status, purchases, and credit card type of its customers in making decisions.

The pharmacy *knows* the pregnancy status of its customers (e.g., via a permitted information flow from a doctor's office to the pharmacy with prescription information). However, it is restricted in how it may *use* this information to protect patient privacy. For example, it may legitimately use pregnancy status directly to dispense medicine, but is prohibited from using pregnancy status to target ads. Indeed, this form of use restriction to protect privacy is embodied in privacy laws like the HIPAA Privacy Rule [101] and in many corporate policies (e.g., [56, 93]). They reflect the understanding that knowledge restrictions are inadequate to protect privacy in settings where the knowledge of certain information types may be used to achieve certain desired purposes (e.g., treatment) but not for others (e.g., marketing). Prior work provides methods for enforcing these explicit use restrictions in human and automated decision-making systems (e.g., [117, 136, 137]).

Use restrictions get more nuanced when proxy use comes into play. For example, instead of using the pregnancy status information available to the pharmacy, the retail store could use information in the purchase history that are strong predictors (or proxies) for pregnancy status (e.g., pre-natal vitamins) to target ads. Our goal is to capture this form of proxy use.

In [35], we propose a definition of proxy use that is an instance of our general template of interpretation of causally influential internal factors. Informally, a program exhibits proxy use of a protected attribute if there exists an internal computation such that the internal computation is 1) a proxy, that is, it is strongly associated with the protected information type, and is 2) used, that it is are causally influential on the outcome.

We then use this definition of proxy use as a building block to construct a theory of

*use privacy* and *proxy non-discrimination*. In this chapter, we first describe a definition for proxy use in Section 6.3. The development of this definition is guided by axioms that characterize reasonable conditions for proxy use. We then describe theories of use privacy (Section 6.1) and proxy non-discrimination (Section 6.2) using the definition of proxy use as a building block. We evaluate the performance of the detection algorithm and show that, in particular cases, the runtime of our system scales linearly in the size of the model.

## 6.1 Use Privacy

We return to the Target example described earlier in the chapter to motivate our notion of use privacy. Historically, data collected in a context of interaction between a retailer and a consumer is not expected to result in flows of health information. However, such flow constraints considered in significant theories of privacy (e.g., see Nissenbaum [97]) cannot be enforced because of possible statistical inferences. In particular, prohibited information types (e.g., pregnancy status) could be inferred from legitimate flows (e.g., shopping history). Thus, the theory of use privacy instead ensures that the data processing systems "simulate ignorance" of protected information types (e.g., pregnancy status) and their proxies (e.g., purchase history) by not using them in their decision-making. Because not all instances of proxy use of a protected information type are inappropriate, our theory of use privacy makes use of a normative judgment oracle that makes this inappropriateness determination for a given instance.

We model the personal data processing system as a program $p$. The use privacy constraint governs a protected information type $Z$. Our definition of use privacy makes use of two building blocks: (1) a function that given $p$, $Z$, and a population distribution $\mathcal{P}$ returns a witness $w$ of proxy use of $Z$ in a program $p$ (if it exists); and (2) a normative judgment oracle $\mathcal{O}(w)$ that given a specific witness returns a judgment on whether the specific proxy use is appropriate (TRUE) or not (FALSE).

**Definition 12** (Use Privacy)**.** *Given a program $p$, protected information type $Z$, normative judgment oracle $\mathcal{O}$, and population distribution $\mathcal{P}$, use privacy in a program $p$ is violated if there exists a witness $w$ in $p$ of proxy use of $Z$ in $\mathcal{P}$ such that $\mathcal{O}(w)$ returns* FALSE.

In this work, we formalize the computational component of the above definition of use privacy, by using our definition of proxy use which formalizes what it means to use an information type directly or through proxies and design an algorithm to detect proxy uses in programs. We assume that the normative judgment oracle is given to us and use it to identify inappropriate proxy uses and then repair them.

This definition cleanly separates computational considerations that are automatically enforceable and ethical judgments that require input from human experts. This form of separation exists also in some prior work on privacy [53] and fairness [45].

## 6.2 Proxy Non-discrimination

The definition of privacy presented above can also be viewed as characterizing non-discrimination when the protected information type is the membership in a certain class. We briefly discuss how this notion of non-discrimination relates existing notions in the law.

The theory of proxy non-discrimination prohibits the proxy use of membership in a protected class for certain decisions. Currently, such restrictions for protected classes based on gender or race are required by the law for education, credit, and employment. Indeed, our treatment of proxy use combines elements of causation and association found in two different parts of anti-discrimination law in the US adapted to the setting of automated decision making systems.

Title VII of U.S. Civil Rights Act prohibits use of race, sex, and other protected attributes for employment decisions [3]. Similar laws govern credit [49] and housing decisions [73]. The case law on enforcing these laws has developed various definitions of when such a protected attribute is *used* for a decision.

Direct *disparate treatment*, on the one hand, corresponds to the obvious case: purposefully and directly using the value of a person's race or sex as an input to a decision-making process, a causal property of the process given that data is unlikely to be provided accidentally to such a process. *Disparate impact*, on the other hand, occurs when the same rule is applied to the protected class without regard for class membership but results in significantly worse outcomes for that class. The courts and regulators have used a variety of heuristics and statistical methods to define "significantly". The most well known is 80% rule, which requires that the rate of hiring of a protected class should be within 80% of the rest [142]. These significance tests each measure the degree of association, but not necessarily causation, between membership in a protected class and employment outcomes.

The courts also recognize more subtle indirect usage, such as pretextually using neighborhood (redlining) or education level as a proxy for race [108]. nalogous to use privacy, our definition of proxy use allows for the formalization of such indirect uses. A further complication is that employers can defend themselves against disparate impact by showing that the difference arose due to a *business necessity*. For example, a moving company may require employees be able to lift 200lbs, a requirement yielding a disparate impact on women, but possibly justifiable as a business necessity. Thus, for automated testing of disparate impact to scale, it will require some method of screening out suspected cases that are justified by a business necessity. Similarly, the theory of disparate treatment contains exceptions for *bona fide occupational qualifications* for gender.

Consider a case of external auditors discovering associations between race and outcomes in a system that predicts the risk of recidivism used in sentencing systems (as in Angwin et al. [8]). Using a theory of proxy discrimination will allow an analyst to identify proxies that explain the presence of these associations, and then make fine-grained judgements of whether the use of these proxies is justified for predicting recidivism.

## 6.3 Proxy Use

We now present an axiomatically justified, formal definition of proxy use in data-driven programs. Our definition for proxy use of a protected information type involves *decomposing* a program to find an intermediate computation whose result exhibits two properties:

- *Proxy*: strong association with the protected type
- *Use*: causal influence on the output of the program

In § 6.3.1, we present a sequence of examples to illustrate the challenge in identifying proxy use in systems that operate on data associated with a protected information type. In doing so, we will also contrast our work with closely-related work in privacy and fairness. In §6.3.2, we formalize the notions of proxy and use, preliminaries to the definition. The definition itself is presented in §6.3.3 and §6.3.4. Finally, in §6.3.5, we provide an axiomatic characterization of the notion of proxy use that guides our definitional choices.

### 6.3.1 Examples of Proxy Use

Prior work on detecting use of protected information types [32, 50, 80, 135] and leveraging knowledge of detection to eliminate inappropriate uses [50] have treated the system as a black-box. Detection relied either on experimental access to the black-box [32, 80] or observational data about its behavior [50, 135]. Using a series of examples motivated by the Target case, we motivate the need to peer inside the black-box to detect proxy use.

**Example 1.** *(Explicit use, Fig. 6.1a) A retailer explicitly uses pregnancy status from prescription data available at its pharmacy to market baby products.*

This form of explicit use of a protected information type can be discovered by existing black-box experimentation methods that establish causal effects between inputs and outputs (e.g., see [32, 80]).

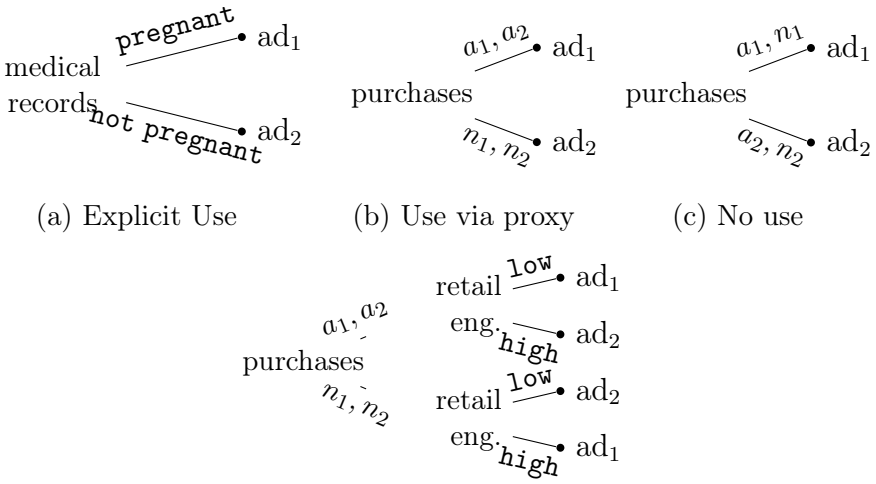
**Example 2.** *(Inferred use, Fig. 6.1b) Consider a situation where purchase history can be used to accurately predict pregnancy status. A retailer markets specific products to individuals who have recently purchased products indicative of pregnancy (e.g., $a_1, a_2 \in$ purchases).*

This example, while very similar in effect, does not use health information directly. Instead, it infers pregnancy status via associations and then uses it. Existing methods (see [50, 135]) can detect such associations between protected information types and outcomes in observational data.

**Example 3.** *(No use, Fig. 6.1c) Retailer uses some uncorrelated selection of products ($a_1, n_1 \in$ purchases) to suggest ads.*

In this example, even though the retailer could have inferred pregnancy status from the purchase history, no such inference was used in marketing products. As associations are commonplace, a definition of use disallowing such benign use of associated data would be too restrictive for practical enforcement.

**Example 4.** *(Masked proxy use, Fig. 6.1d) Consider a more insidious version of Example 2. To mask the association between the outcome and pregnancy status, the company*




(a) Explicit Use (b) Use via proxy (c) No use

(d) Masked use via proxy

Figure 6.1: Examples of models (decision trees) used by a retailer for offering medicines and for selecting advertisements to show to customers. The retailer uses pregnancy status, past purchases, and customer's level of retail engagement. Products $a_1$ and $a_2$ are associated with pregnancy (e.g., prenatal vitamins, scent-free lotions) whereas products $n_1$ and $n_2$ are associated with a lack of pregnancy (e.g., alcohol, camping gear); all four products are equally likely. Retail engagement, (`high` or `low`), indicating whether the customer views ads or not, is independent of pregnancy.

*also markets baby products to people who are not pregnant, but have low retail engagement, so these advertisements would not be viewed in any case.*

While there is no association between pregnancy and outcome in both Example 3 and Example 4, there is a key difference between them. In Example 4, there is an intermediate computation based on aspects of purchase history that is a predictor for pregnancy status, and this predictor is used to make the decision, and therefore is a case of proxy use. In contrast, in Example 3, the intermediate computation based on purchase history is uncorrelated with pregnancy status. Distinguishing between these examples by measuring associations using black box techniques is non-trivial. Instead, we leverage white-box access to the code of the classifier to identify the intermediate computation that serves as a proxy for pregnancy status. Precisely identifying the particular proxy used also aids the normative decision of whether the proxy use is appropriate in this setting.

### 6.3.2 Notation and Preliminaries

We assume individuals are drawn from a population distribution $\mathcal{P}$, in which our definitions are parametric. Random variables $W, X, Y, Z, \ldots$ are functions over $\mathcal{P}$, and the notation $W \in \mathcal{W}$ represents that the type of random variable is $W : \mathcal{P} \to \mathcal{W}$. An important random variable used throughout this chapter is $\mathbf{X}$, which represents the vector of features of an individual that is provided to a predictive model. A predictive model is denoted by

| | |
|---|---|
| $f$ | A function |
| $\langle \mathbf{X}, \mathcal{A} \rangle_{\mathcal{P}}$ | A model, which is a function $\mathcal{A}$ used for prediction, operating on random variables $\mathbf{X}$, in population $\mathcal{P}$ |
| $X$ | A random variable |
| $p$ | A program |
| $\langle \mathbf{X}, p \rangle_{\mathcal{P}}$ | A syntactic model, which is a program $p$, operating on random variables $\mathbf{X}$ |
| $[p_1/X]p_2$ | A substitution of $p_1$ in place of $X$ in $p_2$ |
| $\mathbf{X}$ | A sequence of random variables |

Table 6.1: Summary of notation used in the chapter

$\langle \mathbf{X}, \mathcal{A} \rangle_{\mathcal{P}}$, where $\mathcal{A}$ is a function that operates on $\mathbf{X}$. For simplicity, we assume that $\mathcal{P}$ is discrete, and that models are deterministic. Table 6.1 summarizes all the notation used in this chapter, in addition to the notation for programs that is introduced later in the chapter.

### Proxies

A *proxy* for a random variable $Z$ is a random variable $X$ that is perfectly correlated with $Z$. Informally, it is possible to use $X$ and $Z$ interchangeably in any function drawing inputs from the same distribution.

**Definition 13** (Perfect Proxy)**.** *A random variable $X \in \mathcal{X}$ is a* perfect proxy for $Z \in \mathcal{Z}$ *if there exist functions $f : \mathcal{X} \to \mathcal{Z}, g : \mathcal{Z} \to \mathcal{X}$, such that* $\Pr(Z = f(X)) = \Pr(g(Z) = X) = 1$. □

While this notion of a proxy is too strong in practice, it is useful as a starting point to explain the key ideas in our definition of proxy use. This definition captures two key properties of proxies, *two-sidedness* and *invariance under renaming*.

**Two-sidedness** Definition 13 captures the property that proxies admit predictors in both directions: it is possible to construct a predictor of $X$ from $Z$, and vice versa. This strict two-sided association criterion distinguishes benign use of associated information from proxy use as illustrated in the following example.

**Example 5.** *Recall that in Figure 6.1, $a_1, a_2$ is a proxy for pregnancy status. In contrast, consider Example 3, where purchase history is an influential input to the program that serves ads to. Suppose that the criteria is to serve ads to those with $a_1, n_1$ in their purchase history. According to Definition 13, neither purchase history or $a_1, n_1$ are proxies, because pregnancy status does not predict purchase history or $a_1, n_1$. However, if Definition 13 were to allow one-sided associations, then purchase history would be a proxy because it can predict pregnancy status. This would have the unfortunate effect of implying that the benign application in Example 3 has proxy use of pregnancy status.* □

**Invariance under renaming** This definition of a proxy is invariant under renaming of the values of a proxy. Suppose that a random variable evaluates to 1 when the protected information type is 0 and vice versa, then this definition still identifies the random variable as a proxy.

#### Influence

Our definition of influence aims to capture the presence of a causal dependence between a variable and the output of a function. Intuitively, a variable $x$ is influential on $f$ if it is possible to change the value of $f$ by changing $x$ while keeping the other input variables fixed.

**Definition 14.** *For a function $f(x, y)$, $x$ is influential if and only if there exists values $x_1$, $x_2$, $y$, such that $f(x_1, y) \neq f(x_2, y)$.* □

In Figure 6.1a, pregnancy status is an influential input of the system, as just changing pregnancy status while keeping all other inputs fixed changes the prediction. Influence, as defined here, is identical to the notion of interference used in the information flow literature.

### 6.3.3 Definition

We use an abstract framework of program syntax to reason about programs without specifying a particular language to ensure that our definition remains general. Our definition relies on syntax to reason about decompositions of programs into intermediate computations, which can then be identified as instances of proxy use using the concepts described above.

**Program decomposition** We assume that models are represented by programs. For a set of random variables $\mathbf{X}$, $\langle \mathbf{X}, p \rangle_{\mathcal{P}}$ denotes the assumption that $p$ will run on the variables in $\mathbf{X}$. Programs are given meaning by a denotation function $[\![\cdot]\!]_{\mathbf{X}}$ that maps programs to functions. If $\langle \mathbf{X}, p \rangle_{\mathcal{P}}$, then $[\![p]\!]$ is a function on variables in $\mathbf{X}$, and $[\![p]\!](\mathbf{X})$ represents the random variable of the outcome of $p$, when evaluated on the input random variables $\mathbf{X}$. Programs support substitution of free variables with other programs, denoted by $[p_1/X]p_2$, such that if $p_1$ and $p_2$ programs that run on the variables $\mathbf{X}$ and $\mathbf{X}, X$, respectively, then $[p_1/X]p_2$ is a program that operates on $\mathbf{X}$.

A decomposition of program $p$ is a way of rewriting $p$ as two programs $p_1$ and $p_2$ that can be combined via substitution to yield the original program.

**Definition 15** (Decomposition)**.** *Given a program $p$, a decomposition $(p_1, X, p_2)$ consists of two programs $p_1$, $p_2$, and a fresh variable $X$, such that $p = [p_1/X]p_2$.* □

For the purposes of our proxy use definition we view the first component $p_1$ as the intermediate computation suspected of proxy use, and $p_2$ as the rest of the computation that takes in $p_1$ as an input.

**Definition 16** (Influential Decomposition)**.** *Given a program $p$, a decomposition $(p_1, X, p_2)$ is influential iff $X$ is influential in $p_2$.* □

**Main definition**

**Definition 17** (Proxy Use)**.** *A program $\langle \mathbf{X}, p \rangle_{\mathcal{P}}$ has proxy use for a random variable $Z$ if there exists an influential decomposition $(p_1, X, p_2)$ of $\langle \mathbf{X}, p \rangle_{\mathcal{P}}$, and $[\![p_1]\!](\mathbf{X})$ is a proxy for $Z$.* □

**Example 6.** *In Figure 6.1d, this definition would identify proxy use using the decomposition $(p_1, U, p_2)$, where $p_2$ is the entire tree, but with the condition $(a_1, a_2 \in \textit{purchases})$ replaced by the variable $U$. In this example, $U$ is influential in $p_2$, since changing the value of $U$ changes the outcome. Also, we assumed that the condition $(a_1, a_2 \in \textit{purchases})$ is a perfect predictor for pregnancy, and is therefore a proxy for pregnancy. Therefore, according to our definition of proxy use, the model in 6.1d has proxy use of pregnancy status.*

### 6.3.4 A Quantitative Relaxation

Definition 17 is too strong in one sense and too weak in another. It requires that intermediate computations be perfectly correlated with a protected attribute, and that there exists *some* input, however improbable, in which the result of the intermediate computation is relevant to the model. For practical purposes, we would like to capture imperfect proxies that are strongly associated with an attribute, but only those whose influence on the final model is appreciable. To relax the requirement of perfect proxies and non-zero influence, we quantify these two notions to provide a parameterized definition.

**$\epsilon$-proxies** We wish to measure how strongly a random variable $X$ is a proxy for a random variable $Z$. Recall the two key requirements from the earlier definition of a proxy: (i) the association needs to be two-sided, and (ii) the association needs to be invariant under renaming of the random variables. The *variation of information metric* $d_{\text{var}}(X, Z) = H(X|Z) + H(Z|X)$ [29] is one measure that satisfies these two requirements. The first component in the metric, the conditional entropy of $X$ given $Z$, $H(X|Z)$, measures how well $X$ can be predicted from $Z$, and $H(Z|X)$ measures how well $Z$ can be predicted from $X$, thus satisfying the requirement for the metric being two-sided. Additionally, one can show that conditional entropies are invariant under renaming, thus satisfying our second criteria. To obtain a normalized measure in $[0, 1]$, we choose $1 - \frac{d_{\text{var}}(X,Z)}{H(X,Z)}$ as our measure of association, where the measure being 1 implies perfect proxies, and 0 implies statistical independence. Interestingly, this measure is identical to normalized mutual information [29], a standard measure that has also been used in prior work in identifying associations in outcomes of machine learning models [135].

**Definition 18** (Proxy Association)**.** *Given two random variables $X$ and $Z$, the strength of a proxy is given by normalized mutual information,*

$$d(X, Z) = 1 - \frac{H(X|Z) + H(Z|X)}{H(X, Z)}$$

*where $X$ is defined to be an $\epsilon$-proxy for $Z$ if $d(X, Z) \geq \epsilon$.*

**$\delta$-influential decomposition** Recall that for a decomposition $(p_1, X, p_2)$, in the qualitative sense, influence is given by interference which implies that there exists $x$, $x_1$, $x_2$, such that $[\![p_2]\!](x_1, x) \neq [\![p_2]\!](x_2, x)$. Here $x_1$, $x_2$ are values for the output of $p_1$, that for a given $x$, change the outcome of $p_2$. However, this definition is too strong as it requires only a single pair of values $x_1$, $x_2$ to show that the outcome can be changed by $p_1$ alone. To measure influence, we quantify interference by using Quantitative Input Influence (QII), a causal measure of input influence introduced in [34]. In our context, for a decomposition $(p_1, X, p_2)$, the influence of $p_1$ on $p_2$ is given by:

$$\iota(p_1, p_2) = \mathbb{E}_{\mathbf{X}, \mathbf{X}' \overset{\$}{\leftarrow} \mathcal{P}}([\![p]\!](\mathbf{X}) \neq [\![p_2]\!](\mathbf{X}, [\![p_1]\!](\mathbf{X}'))).$$

Intuitively, this quantity measures the likelihood of finding randomly chosen values of the output of $p_1$ that would change the outcome of $p_2$.

**Definition 19** (Decomposition Influence)**.** *Given a decomposition $(p_1, X, p_2)$, the influence of the decomposition is given by the QII of $X$ on $p_2$. A decomposition $(p_1, X, p_2)$ is defined to be $\delta$-influential if $\iota(p_1, p_2) > \delta$.*

**$(\epsilon, \delta)$-proxy use** Now that we have quantitative versions of the primitives used in Definition 17, we are in a position to define quantitative proxy use (Definition 20). The structure of this definition is the same as before, with quantitative measures substituted in for the qualitative assertions used in Definition 17.

**Definition 20** ($(\epsilon, \delta)$-proxy use)**.** *A program $\langle \mathbf{X}, p \rangle_{\mathcal{P}}$ has $(\epsilon, \delta)$-proxy use of random variable $Z$ iff there exists a $\delta$-influential decomposition $(p_1, X, p_2)$, such that $[\![p]\!](\mathbf{X})$ is an $\epsilon$-proxy for $Z$.*

This definition is a strict relaxation of Definition 17, which reduces to $(1, 0)$-proxy use.

### 6.3.5 Axiomatic Basis for Definition

We now motivate our definitional choices by reasoning about a natural set of properties that a notion of proxy use should satisfy. We first prove an important impossibility result that shows that no definition of proxy use can satisfy four natural semantic properties of proxy use. The central reason behind the impossibility result is that under a purely semantic notion of function composition, the causal effect of a proxy can be made to disappear. Therefore, we choose a syntactic notion of function composition for the definition of proxy use presented above. The syntactic definition of proxy use is characterized by syntactic properties which map very closely to the semantic properties.

**Property 1.** (Explicit Use) *If $Z$ is an influential input of the model $\langle \{\mathbf{X}, Z\}, \mathcal{A} \rangle_{\mathcal{P}}$, then $\langle \{\mathbf{X}, Z\}, \mathcal{A} \rangle_{\mathcal{P}}$ has proxy use of $Z$.*

This property identifies the simplest case of proxy use: if an input to the model is influential, then the model exhibits proxy use of that input.

**Property 2.** (Preprocessing) *If a model $\langle \{\mathbf{X}, X\}, \mathcal{A} \rangle_{\mathcal{P}}$ has proxy use of random variable $Z$, then for any function $f$ such that $\Pr(f(\mathbf{X}) = X) = 1$, let $\mathcal{A}'(x) = \mathcal{A}(x, f(x))$. Then, $\langle \mathbf{X}, \mathcal{A}' \rangle_{\mathcal{P}}$ has proxy use of $Z$.*

This property covers the essence of proxy use where instead of being provided a protected information type explicitly, the program uses a strong predictor for it instead. This property states that models that use inputs explicitly and via proxies should not be differentiated under a reasonable theory of proxy use.

**Property 3.** (Dummy) *Given* $\langle \mathbf{X}, \mathcal{A} \rangle_{\mathcal{P}}$*, define* $\mathcal{A}'$ *such that for all* $x, x'$*,* $\mathcal{A}'(x, x') = \mathcal{A}(x)$*, then* $\langle \mathbf{X}, \mathcal{A} \rangle_{\mathcal{P}}$ *has proxy use for some* $Z$ *iff* $\langle \{\mathbf{X}, X\}, \mathcal{A}' \rangle_{\mathcal{P}}$ *has proxy use of* $Z$*.*

This property states that the addition of an input to a model that is not influential, i.e., has no effect on the outcomes of the model, has no bearing on whether a program has proxy use or not. This property is an important sanity check that ensures that models aren't implicated by the inclusion of inputs that they do not use.

**Property 4.** (Independence) *If* $\mathbf{X}$ *is independent of* $Z$ *in* $\mathcal{P}$*, then* $\langle \mathbf{X}, \mathcal{A} \rangle_{\mathcal{P}}$ *does not have proxy use of* $Z$*.*

Independence between the protected information type and the inputs ensures that the model cannot infer the protected information type for the population $\mathcal{P}$. This property captures the intuition that if the model cannot infer the protected information type then it cannot possibly use it.

While all of these properties seem intuitively desirable, it turns out that these properties can not be achieved simultaneously.

**Theorem 10.** *No definition of proxy use can satisfy Properties 1-4 simultaneously.*

*Proof.* Proof by contradiction. Assume that there exists a definition of proxy usage that satisfies all four properties. Let $\mathcal{X} = \{0, 1\}$, and $X$ is a uniform Bernoulli variable over $\mathcal{X}$. The model $\mathcal{A}(x) = x$ is the identity function. Let $Z$ be an independent uniform Bernoulli variable. According to (independence), $\mathcal{A}$ has no proxy usage of $Z$. Choose $\mathcal{A}'(x, z) = \mathcal{A}(x)$ which operates over $\mathcal{X} \times \mathcal{Z}$. By (dummy), $\mathcal{A}'$ has no implicit use of $Z$. We choose the following bijective transformation: $f(x, z) = (u, z) = (x \oplus z, z)$, and $f^{-1}(u, z) = (u \oplus z, z)$ In this transformed space, we choose $\mathcal{A}'' = \mathcal{A}' \circ f^{-1}$. Therefore, $\mathcal{A}''(u, z) = u \oplus z$, since $\mathcal{A}''(u, z) = \mathcal{A}'(f^{-1}(u, z)) = \mathcal{A}'(u \oplus z, z) = u \oplus z$. According to (representation independence), $\mathcal{A}''$ has no implicit use of $Z$. However, since $z$ is an influential input of the model, according to (explicit use of proxy), $\mathcal{A}''$ has implicit use of $Z$. Therefore, we have a contradiction. □

The key intuition behind this result is that Property 2 requires proxy use to be preserved when an input is replaced with a function that predicts that input via composition. However, with a purely semantic notion of function composition, after replacement, the proxy may get canceled out. To overcome this impossibility result, we choose a more syntactic notion of function composition, which is tied to how the function is represented as a program, and looks for evidence of proxy use within the representation.

We now proceed to the axiomatic justification of our definition of proxy use. As in our attempt to formalize a semantic definition, we base our definition on a set of natural properties given below. These are syntactic versions of their semantic counterparts defined earlier.

**Property 5.** (Syntactic Explicit Use) *If* $X$ *is a proxy of* $Z$*, and* $X$ *is an influential input of* $\langle \{\mathbf{X}, X\}, p \rangle_{\mathcal{P}}$*, then* $\langle \{\mathbf{X}, X\}, p \rangle_{\mathcal{P}}$ *has proxy use.*

**Algorithm 2** Detection for expression programs.

**Require:** association ($d$), influence($\iota$) measures
  **procedure** ProxyDetect($p, \mathbf{X}, Z, \epsilon, \delta$)
    $P \leftarrow \varnothing$
    **for** each subprogram $p_1$ appearing in $p$ **do**
      **for** each program $p_2$ such that $[p_2/u]p_1 = p$ **do**
        **if** $\iota(p_1, p_2) \geq \delta \wedge d([\![p_1]\!](\mathbf{X}), Z) \geq \epsilon$ **then**
          $P \leftarrow P \cup \{(p_1, p_2)\}$
        **end if**
      **end for**
    **end for**
    **return** $P$
  **end procedure**

**Property 6.** (Syntactic Preprocessing) *If* $\langle\{\mathbf{X}, X\}, p_1\rangle_{\mathcal{P}}$ *has proxy use of* $Z$*, then for any* $p_2$ *such that* $\Pr([\![p_2]\!](\mathbf{X}) = X) = 1$*,* $\langle\mathbf{X}, [p_2/X]p_1\rangle_{\mathcal{P}}$ *has proxy use of* $Z$*.*

**Property 7.** (Syntactic Dummy) *Given a program* $\langle\mathbf{X}, p\rangle_{\mathcal{P}}$*,* $\langle\mathbf{X}, p\rangle_{\mathcal{P}}$ *has proxy use for some* $Z$ *iff* $\langle\{\mathbf{X}, X\}, p\rangle_{\mathcal{P}}$ *has proxy use of* $Z$*.*

**Property 8.** (Syntactic Independence) *If* $\mathbf{X}$ *is independent of* $Z$*, then* $\langle\mathbf{X}, p\rangle_{\mathcal{P}}$ *does not have proxy use of* $Z$*.*

Properties 5 and 6 together characterize a complete inductive definition, where the induction is over the structure of the program. Suppose we can decompose programs $p$ into $(p_1, X, p_2)$ such that $p = [p_1/X]p_2$. Now if $X$, which is the output of $p_1$, is a proxy for $Z$ and is influential in $p_2$, then by Property 5, $p_2$ has proxy use. Further, since $p = [p_1/X]p_2$, by Property 6, $p$ has proxy use. This inductive definition where we use Property 5 as the base case and Property 6 for the induction step, precisely characterizes Definition 17. Additionally, it is can be shown that Definition 17 also satisfies Properties 7 and 8. Essentially, by relaxing our notion of function composition to a syntactic one, we obtain a practical definition of proxy use characterized by the natural axioms above.

## 6.4 Detecting Proxy Use

In this section, we present an algorithm for identifying proxy use of specified variables in a given machine-learning model (Algorithm 2, Appendix A.1 contains a more formal presentation of the algorithm for the interested reader). The algorithm is program-directed and is directly inspired by the definition of proxy use in the previous section. We prove that the algorithm is complete in a strong sense — it identifies every instance of proxy use in the program (Theorem 12). We also describe three optimizations that speed up the detection algorithm: sampling, reachability analysis, and contingency tables.

### 6.4.1 Environment Model

The environment in which our detection algorithm operates is comprised of a data processor, a dataset that has been partitioned into analysis and validation subsets, and a machine learning model trained over the analysis subset. We assume that the data processor does not act to evade the detection algorithm, and the datasets correspond to a representative sample from the population we wish to test proxy use with respect to. Additionally, we assume that information types we wish to detect proxies of are also part of the validation data. We discuss these points further in Section 6.9.

For the rest of this chapter we focus on an instance of the proxy use definition, where we assume that programs are written in the simple expression language shown in Figure 6.2. However, our techniques are not tied to this particular language, and the key ideas behind them apply generally. This language is rich enough to support commonly-used models such as decision trees, linear and logistic regression, Naive Bayes, and Bayesian rule lists. Programs are functions that evaluate arithmetic terms, which are constructed from real numbers, variables, common arithmetic operations, and if-then-else ($\textbf{ite}(\cdot,\cdot,\cdot)$) terms. Boolean terms, which are used as conditions in **ite** terms, are constructed from the usual connectives and relational operations. Finally, we use $\lambda$-notation for functions, i.e., $\lambda x.e$ denotes a function over $x$ which evaluates $e$ after replacing all instances of $x$ with its argument. Details on how machine learning models such as linear models, decision trees, and random forests are translated to this expression language are discussed in Appendix A.1.2 and consequences of the choice of language and decomposition in that language are further discussed in more detail in Section 6.9.

**Distributed proxies** Our use of program decomposition provides for partial handling of *distributed representations*, the idea that concepts can be distributed among multiple entities. In our case, influence and association of a protected information type can be distributed among multiple program points. First, substitution (denoted by $[p_1/X]p_2$) is defined to replace *all* instances of variable $X$ in $p_2$ with the program $p_1$. If there are multiple instances of $X$ in $p_2$, they are still describing a single decomposition and thus the multiple instances of $p_2$ in $p_1$ are viewed as a single proxy. Further, implementations of substitution can be (and is in our implementation) associativity-aware: programs like $x_1 + x_2 + x_3$ can be equivalent regardless of the order of the expressions in that they can be decomposed in exactly the same set of ways. If a proxy is distributed among $x_1$ and $x_3$, it will still be considered by our methods because $x_1 + (x_2 + x_3)$ is equivalent to $(x_1 + x_3) + x_2$, and the sub-expression $x_1 + x_3$ is part of a valid decomposition. Allowing such equivalences within the implementation of substitution partially addresses the problem that our theory does not respect semantic equivalence, which is a necessary consequence of Theorem 10.

### 6.4.2 Analyzing Proxy Use

Algorithm 2 describes a general technique for detecting $(\epsilon, \delta)$-proxy use in expression programs. In addition to the parameters and expression, it takes as input a description of the

$\langle aexp \rangle ::= \mathbb{R} \mid \text{var} \mid \text{op}(\langle aexp \rangle, \ldots, \langle aexp \rangle)$
$\quad \mid \quad \textbf{ite}(\langle bexp \rangle, \langle aexp \rangle, \langle aexp \rangle)$

$\langle bexp \rangle ::= \text{T} \mid \text{F} \mid \neg \langle bexp \rangle$
$\quad \mid \quad \text{op}(\langle bexp \rangle, \ldots, \langle bexp \rangle)$
$\quad \mid \quad \text{relop}(\langle aexp \rangle, \langle aexp \rangle)$

$\langle prog \rangle ::= \lambda \text{var}_1, \ldots, \text{var}_n \; . \; \langle aexp \rangle$

Figure 6.2: Syntax for the language used in our analysis.

distribution governing the feature variables $X$ and $Z$. In practice this will nearly always consist of an empirical sample, but for the sake of presentation we simplify here by assuming the distribution is explicitly given. In Section 6.7.2, we describe how the algorithm can produce estimates from empirical samples.

The algorithm proceeds by enumerating sub-expressions of the given program. For each sub-expression $e$ appearing in $p$, ProxyDetect computes the set of positions at which $e$ appears. If $e$ occurs multiple times, we consider all possible subsets of occurrences as potential decompositions[1] . It then iterates over all combinations of these positions, and creates a decomposition for each one to test for $(\epsilon, \delta)$-proxy use. Whenever the provided thresholds are exceeded, the decomposition is added to the return set. This proceeds until there are no more subterms to consider. While not efficient in the worst-case, this approach is both sound and complete with respect to Definition 20.

**Theorem 11** (Detection soundness)**.** *Any decomposition* $(p_1, p_2)$ *returned by* ProxyDetect$(p, \mathbf{X}, \epsilon, \delta)$ *is a decomposition of the input program* $p$ *and had to pass the* $\epsilon, \delta$ *thresholds, hence is a* $(\epsilon, \delta)$*-proxy use.*

**Theorem 12** (Detection completeness)**.** *Every decomposition which could be a* $(\epsilon, \delta)$*-proxy use is enumerated by the algorithm. Thus, if* $(p_1, p_2)$ *is a decomposition of* $p$ *with* $\iota(p_1, p_2) \geq d$ *and* $d(\llbracket p_1 \rrbracket(\mathbf{X}), Z) \geq \epsilon$*, it will be returned by* ProxyDetect$(p, \mathbf{X}, \epsilon, \delta)$*.*

Our detection algorithm considers single terms in its decomposition. Sometimes a large number of syntactically different proxies with weak influence might collectively have high influence. A stronger notion of program decomposition that allows a collection of multiple terms to be considered a proxy would identify such a case of proxy use but will have to search over a larger space of expressions. Exploring this tradeoff between scalability and richer proxies is an important topic for future work.

The detection algorithm runs in time $\mathcal{O}\left(|p|\, c\left(|\mathcal{D}| + k\,|\mathcal{D}|\right)\right)$ where $|\mathcal{D}|$ is the size of a dataset employed in the analysis, $c$ is the number of decompositions of a program, $k$ is the maximum number of elements in the ranges of all sub-programs ($|\mathcal{D}|$ in the worst case), and $|p|$ is the number of sub-expressions of a program. The number of decompositions varies from $\mathcal{O}\left(|p|\right)$ to $\mathcal{O}\left(2^{|p|}\right)$ depending on the type of program analyzed. Details can be found in Section 6.7 along with more refined bounds for several special cases.

[1]This occurs often in decision forests (see Figure A.2).

## 6.5 Removing Proxy Use Violations

In this section we present a repair algorithm for removing violations of $(\epsilon, \delta)$-Proxy Use in a model. Our approach has two parts: first (Algorithm 3) is the iterative discovery of proxy uses via the ProxyDetect procedure described in the previous section and second (Algorithm 4) is the repair of the ones found by the oracle to be violations. We describe these algorithms informally here, and Appendix A.2 contains formal descriptions of these algorithms. The iterative discovery procedure guarantees that the returned program is free of violations (Algorithm 6). Our repair procedures operate on the expression language, so they can be applied to any model that can be written in the language. Further, our violation repair algorithm does not require knowledge of the training algorithm that produced the model. The witnesses of proxy use localize where in the program violations occur. To repair a violation we search through expressions *local* to the violation, replacing the one which has the least impact on the accuracy of the model that at the same time reduces the association or influence of the violation to below the $(\epsilon, \delta)$ threshold.

At the core of our violation repair algorithm is the simplification of sub-expressions in a model that are found to be violations. Simplification here means the replacement of an expression that is not a constant with one that is. Simplification has an impact on the model's performance hence we take into account the goal of preserving utility of the machine learning program we repair. We parameterize the procedure with a measure of utility $v$ that informs the selection of expressions and constants for simplification. We briefly discuss options and implementations for this parameter later in this section.

The repair procedure (Algorithm 4) works as follows. Given a program $p$ and a decomposition $(p_1, p_2)$, it first finds the best simplification to apply to $p$ that would make $(p_1, p_2)$ no longer a violation. This is done by enumerating expressions that are *local* to $p_1$ in $p_2$ (Line 3). Local expressions are sub-expressions of $p_1$ as well as $p_1$ itself and if $p_1$ is a guard in an if-then-else expression, then local expressions of $p_1$ also include that if-then-else's true and false branches as well as their sub-expressions. Each of the local expressions corresponds to a decomposition of $p$ into the local expression $p_1'$ and the context around it $p_2'$. For each of these local decompositions we discover the best constant, in terms of utility, to replace $p_1'$ with (Line 4). We then make the same simplification to the original decomposition $(p_1, p_2)$, resulting in $(p_1'', p_2'')$ (Line 5) Using this third decomposition we check whether making the simplification would repair the original violation (Line 6), collecting those simplified programs that do. Finally, we take the best simplification of those found to remove the violation (Line 10). Details on how the optimal constant is selected is described in Appendix A.2.1.

Two important things to note about the repair procedure. First, there is always at least one subprogram on Line 3 that will fix the violation, namely the decomposition $(p_1, p_2)$ itself. Replacing $p_1$ with a constant in this case would disassociate it from the sensitive information type. Secondly, the procedure produces a model that is smaller than the one given to it as it replaces a non-constant expression with a constant. These two let us state the following:

**Algorithm 3** Witness-driven repair.

**Require:** association ($d$), influence ($\iota$), utility ($v$) measures, oracle ($\mathcal{O}$)

**procedure** REPAIR($p, \mathbf{X}, Z, \epsilon, \delta$)
  $P \leftarrow \{d \in \text{PROXYDETECT}(p, \mathbf{X}, Z, \epsilon, \delta) : \text{ not } \mathcal{O}(d)\}$
  **if** $P \neq \emptyset$ **then**
    $(p_1, p_2) \leftarrow$ element of $P$
    $p' \leftarrow \text{PROXYREPAIR}(p, (p_1, p_2), \mathbf{X}, Z, \epsilon, \delta)$
    **return** $\text{REPAIR}(p', \mathbf{X}, Z, \epsilon, \delta)$
  **else**
    **return** $p$
  **end if**
**end procedure**

**Algorithm 4** Local Repair.

**Require:** association ($d$), influence ($\iota$), utility ($v$) measures

**procedure** PROXYREPAIR($p, (p_1, p_2), \mathbf{X}, Z, \epsilon, \delta$)
  $R \leftarrow \{\}$
  **for** each subprogram $p_1'$ of $p_1$ **do**
    $r^* \leftarrow$ Optimal constant for replacing $p_1'$
    $(p_1'', p_2'') \leftarrow (p_1, p_2)$ with $r^*$ subst. for $p_1'$
    **if** $\iota(p_1'', p_2'') \leq \delta \vee d([\![p_1'']\!](\mathbf{X}), Z) \leq \epsilon$ **then**
      $R \leftarrow R \cup [u/r^*]p_2'$
    **end if**
  **end for**
  **return** $\arg\max_{p^* \in R} v(p^*)$
**end procedure**

**Theorem 13.** *Algorithm 3 terminates and returns a program that does not have any* $(\epsilon, \delta)$*-Proxy Use violations (instances of* $(\epsilon, \delta)$*-Proxy Use for which oracle returns false).*

## 6.6 Evaluation

In this section we empirically evaluate our definition and algorithms on several real datasets. In particular, we simulate a financial services application and demonstrate a typical workflow for a practitioner using our tools to detect and repair proxy use in decision trees and linear models (§6.6.1). We highlight that this workflow identifies more proxy uses over a baseline procedure that simply removes features associated with a protected information type. For three other simulated settings on real data sets—contraception advertising, student assistance, and credit advertising—we describe our findings of interesting proxy uses and demonstrate how the outputs of our detection tool would allow a normative judgment oracle to determine the appropriateness of proxy uses (§6.6.2). In §6.6.3, we evaluate the performance of our detection and repair algorithms and show that in particular cases, the

runtime of our system scales linearly in the size of the model. Also, by injecting violations into real data sets so that we have ground truth, we evaluate the completeness of our algorithm, and show a graceful degradation in accuracy as the influence of the violating proxy increases.

**Models and Implementation** Our implementation currently supports linear models, decision trees, random forests, and rule lists. Note that these model types correspond to a range of commonly-used learning algorithms such as logistic regression, support vector machines [27], CART [19], and Bayesian rule lists [82]. Also, these models represent a significant fraction of models used in practice in predictive systems that operate on personal information, ranging from advertising [26], psychopathy [60], criminal justice [15, 16], and actuarial sciences [52, 54]. Our prototype implementation was written in Python, and we use scikit-learn package to train the models used in the evaluation. The benchmarks we describe later in this section were recorded on a Ubuntu Desktop with 4.2 GHz Intel Core i7 and 32GB RAM.

### 6.6.1 Example Workflow

A financial services company would like to expand its client base by identifying potential customers with high income. To do so, the company hires an analyst to build a predictive model that uses age, occupation, education level, and other socio-economic features to predict whether an individual currently has a "high" or "low" income. This practice is in line with the use of analytics in the financial industry that exploit the fact that high-income individuals are more likely to purchase financial products [145].

Because demographic data is known to correlate with marital status [94], the data processor would like to ensure that the trained model used to make income predictions does not effectively infer individuals' marital status from the other demographic variables that are explicitly used. In this context, basing the decision of which clients to pursue on marital status could be perceived as a privacy violation, as other socio-economic variables are more directly related to one's interest and eligibility in various financial services.

To evaluate this scenario, we trained an income prediction model from the UCI Adult dataset which consists of roughly 48,000 rows containing economic and demographic information for adults derived from publicly-available U.S. Census data. One of the features available in this data is marital status, so we omitted it during training, and later used it when evaluating our algorithms. In this scenario, we act as the oracle in order to illustrate the kind of normative judgments an analyst would need to make as an oracle.

After training a classifier on the preprocessed dataset, we found a strong proxy for marital status in terms of an expression involving relationship status. Figure 6.3 visualizes all of the expressions making up the model (marked as •), along with their association and influence measures. In decision trees, sub-expressions like these coincide with decompositions in our proxy use definition; each sub-expression can be associated with a decomposition that cuts out that sub-expression from the tree, and leaves a variable in its place. The connecting lines in the figure denote the sub-expression relationship. Together with the

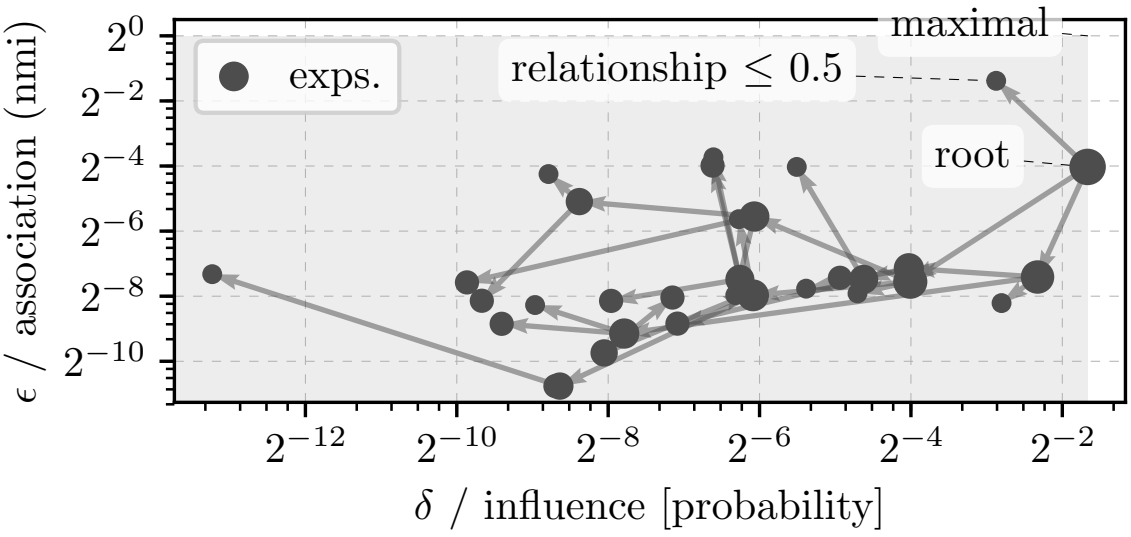



Figure 6.3: The association and influence of the expressions composing a decision tree trained on the UCI Adult dataset. Narrow lines designate the sub-expression relationship. Shaded area designates the feasible values for association and influence between none, and maximal. Marker size denotes the relative size of the sub-expressions pictured.

placement of points on the influence and association scales, this produces an overview of the decision tree and the relationship of its constituent parts to the sensitive attribute.

On further examination the relationship status was essentially a finer-grained version of marital status. While not interesting in itself, this occurrence demonstrates an issue with black-box use of machine learning without closely examining the structure of the data. In particular, one can choose to remove this feature, and the model obtained after retraining will make predictions that have low association with marital status. However, one submodel demonstrated relatively strong proxy use ($\epsilon = 0.1, \delta = 0.1$): `age` $\leq 31$ and `sex` $= 0$ and `capital_loss` $\leq 1882.50$ (labeled A in Figure 6.4). This demonstrates that simply removing a feature does not ensure that proxies are removed. When the model is retrained, the learning algorithm might select new computations over other features to embed in the model, as it did in this example. Also, note that the new proxy combines three additional features. Eliminating all of these features from the data could impact model performance. Instead we can use our repair algorithm to remove the proxy: we designate the unacceptable $\epsilon, \delta$ thresholds (the darkest area in Figure 6.4) and repair any proxies in that range. The result is the decision tree marked with + in the figure. Note that this repaired version has no sub-expressions in the prohibited range and that most of the tree remains unchanged (the • and + markers largely coincide).

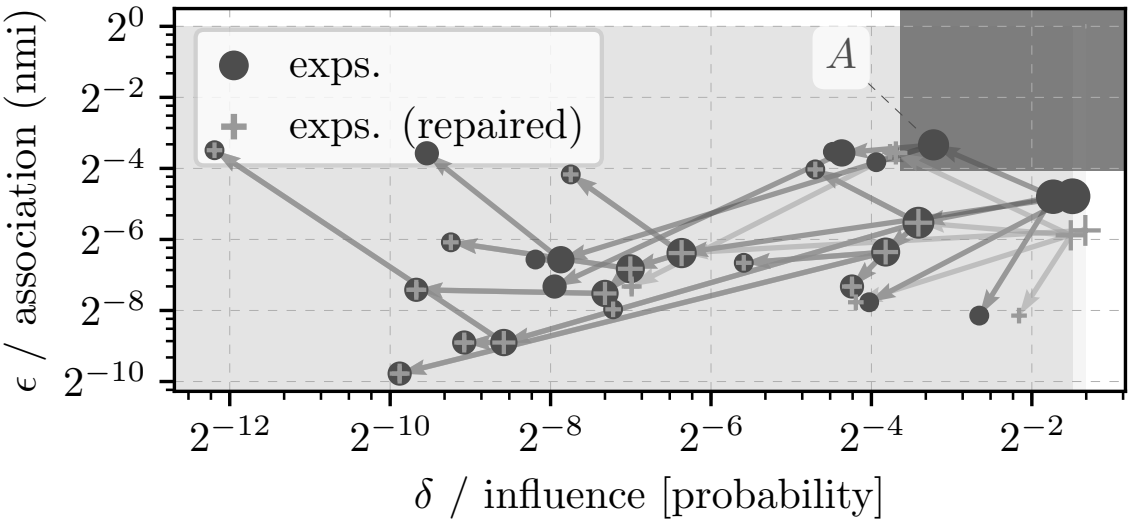



Figure 6.4: Decision tree trained on the UCI Adult dataset but with the `relationship` attribute removed (•), and the repaired version (+) of the same tree. Dark area in the upper-left designates the thresholds used in repair.

### 6.6.2 Other Case Studies

We now discuss interesting examples for proxy use from other case studies, demonstrating how our framework aids normative use privacy judgments.

**Targeted contraception advertising** We consider a scenario in which a data processor wishes to show targeted advertisements for contraceptives to females. To support this goal, the processor collects a dataset from a randomly-sampled female population containing age, level of education, number of children, current employment status, income, level of media exposure, information about the partner's education and occupation, and the type of contraception used by the individual (if any). This dataset is used to train a model that predicts whether an individual uses no contraception, short-term contraception, or long-term contraception. This model is then used to determine who to display advertisements to, under the assumption that individuals who already use short-term contraception are more likely to be receptive to the advertisements.

Because certain religions ban the use of contraception, users belonging to such a religion are on the whole less likely to purchase contraceptives after seeing such an advertisement. The ad-targeting model does not explicitly use a feature corresponding to religion, as this information is not available to the system when ads are displayed. Furthermore, some users may view the use of this information for advertising purposes as a violation of their privacy, so the data processor would like to ensure that the targeting model has not inferred a proxy for this information that is influential in determining whether to show an advertisement.

We evaluated this scenario using data collected for the 1987 National Indonesia Con-

traceptive Survey [5], which contains the features mentioned above, as well as a feature indicating whether the individual's religious beliefs were Islam. To simulate the data processor, we trained a decision tree classifier to predict contraceptive use over all available features except the one corresponding to religion. We then used our detection algorithm to look for a proxy use of religion, using the available data as ground truth to evaluate the effectiveness of our approach.

Although this data is representative of a single country, it illustrates an interesting case of potential use privacy. Our detection algorithm identified the following intermediate computation, which was one of the most influential in the entire model, and the one most closely associated with the religion variable: $ite(\texttt{educ} < 4 \wedge \texttt{nchild} \leq 3 \wedge \texttt{age} < 31, \texttt{no}, \texttt{yes})$. This term predicts that women younger than 31, with below-average education background and fewer than four children will not use contraception. Given that the dataset is comprised entirely of females, closer examination in fact reveals that just the "guard" term $\texttt{educ} < 4$ alone is even more closely associated with religion, and its influence on the model's output is nearly as high. This reveals that the model is using the variable for education background as a proxy for religion, which may be concerning given that this application is focused on advertising.

**Student assistance** A current trend in education is the use of predictive analytics to identify students who are likely to benefit from certain types of interventions to ensure on-time graduation and other benchmark goals [51, 61]. We look at a scenario where a data processor builds a model to predict whether a secondary school student's grades are likely to suffer in the near future, based on a range of demographic features (such as age, gender, and family characteristics), social information (such as involvement in extracurricular activities, amount of reported free time after school), and academic information (e.g., number of reported absences, use of paid tutoring services, intention to continue on to higher education). Based on the outcome of this prediction, the student's academic advisor can decide whether to pursue additional interventions.

Because of the wide-ranging nature of the model's input features, and sensitivity towards the privacy rights of minors, the data processor would like to ensure that the model does not base its decision on inferred facts about certain types of activities that the student might be involved with. For example, alcohol consumption may be correlated with several of the features used by the model, and it may not be seen as appropriate to impose an intervention on a student because their profile suggests that they may have engaged in this activity. Depending on the context in which such an inference were made, the processor would view this as a privacy violation, and attempt to remove it from the model.

To evaluate this scenario, we trained a model on the UCI Student Alcohol Consumption dataset [28]. This data contains approximately 700 records collected from Portuguese public school students, and includes features corresponding to the variables mentioned above. Our algorithm found the following proxy for alcohol use: $ite(\texttt{studytime} < 2 \wedge \texttt{dad\_educ} < 4, \texttt{fail}, ...)$, which predicts that a student who spends at most five hours per week studying, and whose father's level of education is below average, is likely to fail a class in at least one term. Further investigation reveals that $\texttt{studytime} < 2$ was more influential

on the model's output, and nearly as associated with alcohol consumption, as the larger term. This finding suggests that this instance of proxy use can be deemed an appropriate use, and not a privacy violation, as the amount of time a student spends studying is clearly relevant to their academic performance. If instead `dad_educ` $< 4$ alone had turned out to be a proxy use of alcohol consumption, then it may have been a concerning inference about the student's behavior from information about their family history. Our algorithm correctly identified that this is not the case.

**Credit advertisements** We consider a situation where a credit card company wishes to send targeted advertisements for credit cards based on demographic information. In this context, the use of health status for targeted advertising is a legitimate privacy concern [39].

To evaluate this scenario, we trained a model to predict interest in credit cards using the PSID dataset, which contains detailed demographic information for roughly 10,000 families and includes features such as age, employment status income, education, and the number of children. From this, we trained two models: one that identifies individuals with student loans and another that identifies individuals with existing credit cards as the two groups to be targeted.

The first model had a number of instances of proxy use. One particular subcomputation that was concerning was a subtree of the original decision tree that branched on the number of children in the family. This instance provided negative outcomes to individuals with more children, and may be deemed inappropriate for use in this context. In the second model, one proxy was a condition involving income `income` $\leq 33315$. The use of income in this context is justifiable, and therefore this may not be regarded as a use privacy violation.

### 6.6.3 Detection and Repair

For the remainder of the section we focus on evaluating the performance and efficacy of the detection and repair algorithms. We begin by exploring the impact of the dataset and model size on the detection algorithm's runtime.

Figure 6.5 demonstrates the runtime of our detection algorithm on three models trained on the UCI Adult dataset vs. the size of the dataset used for the association and influence computations. The algorithm here was forced to compute the association and influence metrics for each decomposition (normally influence can be skipped if association is below threshold) and thus represents a worst-case runtime. The runtime for the random forest and decision tree scales linearly in dataset size due to several optimizations. The logistic regression does not benefit from these and scales quadratically. Further, runtime for each model scales linearly in the number of decompositions.

Figure 6.6 demonstrates the runtime of the detection algorithm as a function of the number of decompositions of the analyzed model (a proxy for its size). We show two trends in that figure. The black line demonstrates the worst case detection that requires both association and influence computation for each decomposition while the gray line demonstrates the best case where only the association computation is performed. The

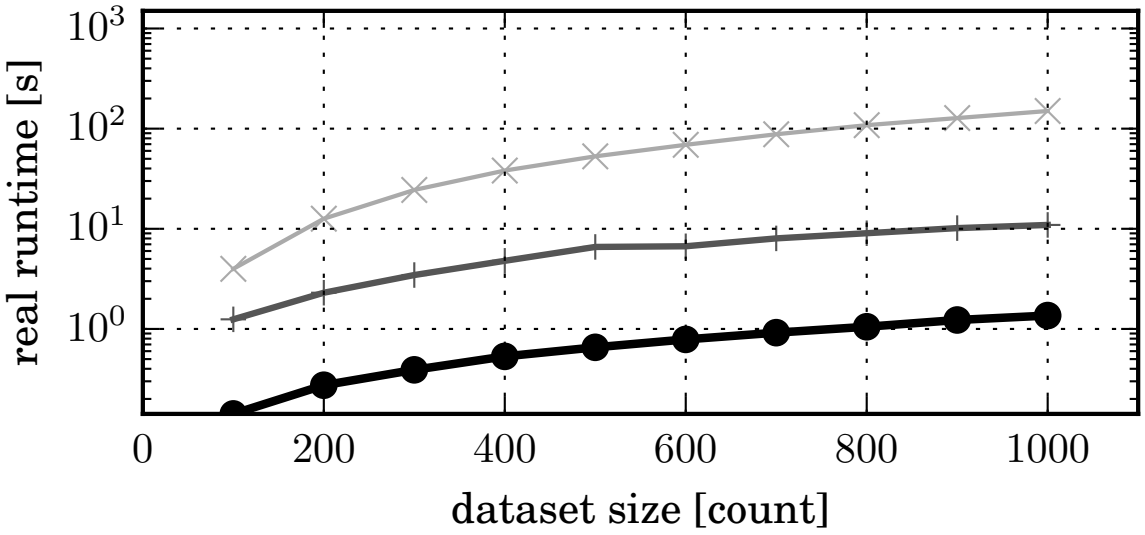



Figure 6.5: Worst-case detection algorithm run-time (average of 5 runs) as a function of input dataset size. Influence and association computed on each decomposition (hence worst-case). The models are decision tree($\circ$), random forest($+$), and logistic regression($\times$) trained on the UCI Adult dataset.

runtime in practice would thus fall somewhere between these two cases, both linear in the number of decompositions. The scalability of our algorithm relative to the number of decompositions has large consequences the its applicability to linear models. We have found that such models, if trained without restricting the number of coefficients, are too big for our sound sub-expression enumeration to handle. In order to deal with this problem we used the Lasso method in order to reduce the number of coefficients in the models we analyzed.

To determine the completeness of our detection algorithm we inserted a proxy in a trained model to determine whether we can detect it. To do this, we used the UCI Student Alcohol Consumption dataset to train two decision trees: one to predict students' grades, and one to predict alcohol consumption. We then inserted the second tree into random positions of the first tree thereby introducing a proxy for alcohol consumption. We observed that in each case, we were able to detect the introduced proxy. While not interesting in itself due to our completeness theorem, we used this experiment to explore how much utility is actually lost due to repair. We evaluate our repair algorithm on a set of similar models with inserted violations of various influence magnitude. The results can be seen in Figure 6.7. We can see that the accuracy (i.e., ratio of instances that have agreement between repaired and unrepaired models) falls linearly with the influence of the inserted proxy. This implies that repair of less influential proxies will incur a smaller accuracy penalty than repair of more influential proxies. In other words, our repair methods do not unduly sacrifice accuracy when repairing only minor violations.

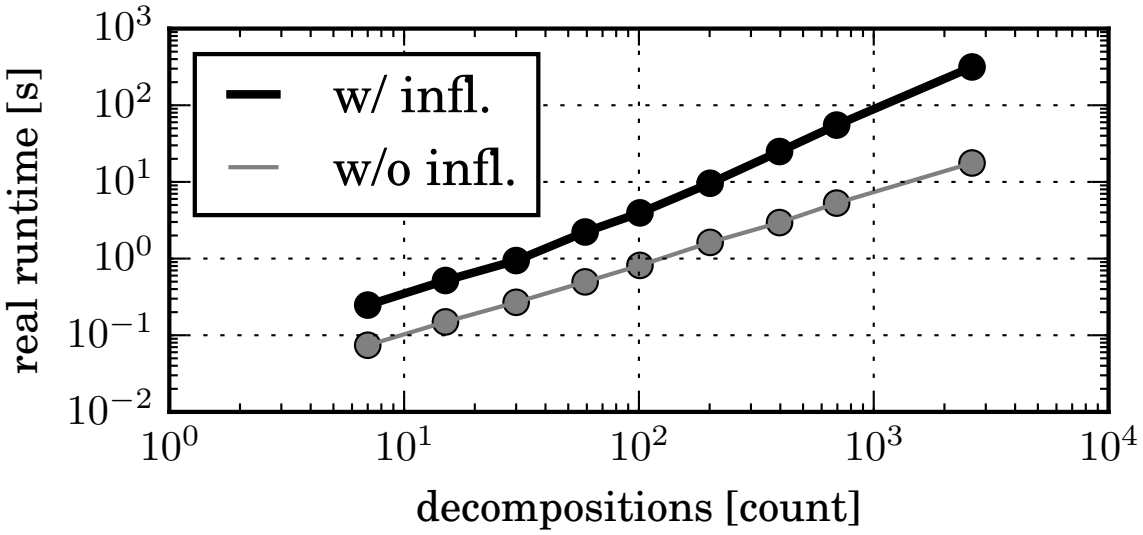



Figure 6.6: Detection algorithm run-time as a function of the number of decompositions in a decision tree trained on the UCI Adult dataset. Detecting violations with $\epsilon = 0.0$ (black dots and line) requires influence computations whereas detecting with $\epsilon = 1.0$ (gray dots and line) does not. A 1000 instance sample of the dataset was used with varying training depths.

A point not well visible in this figure is that occasionally repair incurs no loss of utility. This is due to our use of the scikit-learn library for training decision trees as it does not currently support pruning unnecessary nodes. Occasionally such nodes introduce associations without improving the model's accuracy. These nodes can be replaced by constants without loss. We have also observed this in some of our case studies.

## 6.7 Complexity

The complexity of the presented algorithms depend on several factors, including the type of model being analyzed, the number of elements in the ranges of sub-programs, and reachability of sub-programs by dataset instances . In this section we describe the the complexity characteristics of the detection and repair algorithms under various assumptions. Complexity is largely a property of the association and influence computations and the number of decompositions of the analyzed program. We begin by noting our handling of probability distributions as specified by datasets, several quantities of interest, discuss the complexity of components of our algorithms, and conclude with overall complexity bounds.

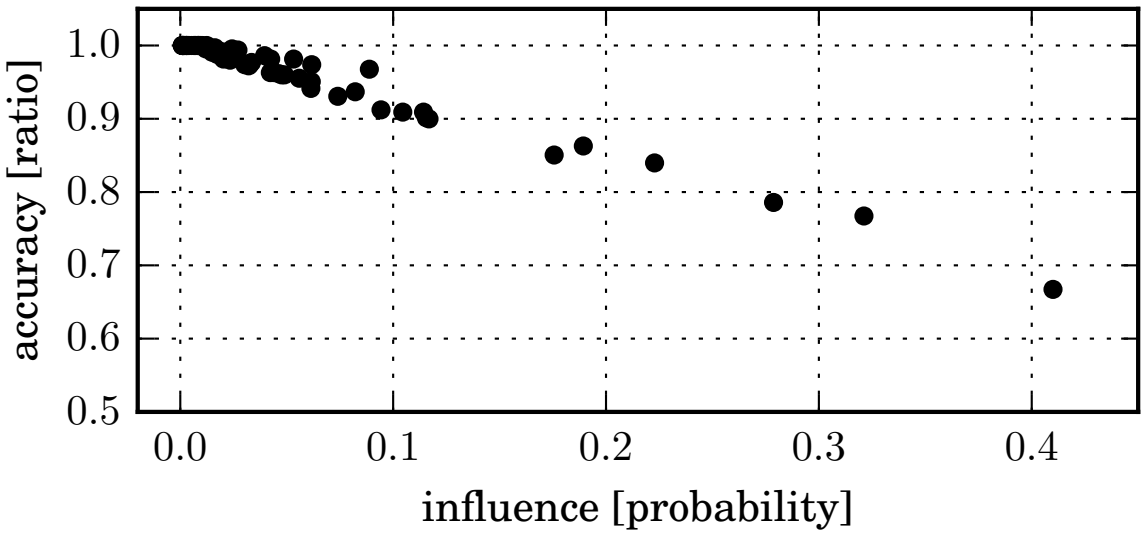



Figure 6.7: Repaired accuracy vs. influence of proxy during repair of a synthetic proxy inserted into random positions of a decision tree trained on the UCI Student Alcohol Consumption dataset. Accuracy is agreement to non-repaired model. The synthetic model is a (1.0)-proxy for alcohol use, inserted into a decision tree predicting student grade. Repair is configured for $(0.01, 0.01)$-proxy use removal. Note that other proxies (if they exist) are not repaired in this experiment.

### 6.7.1 Distributions, datasets, and probability

It is rarely the case that one has access to the precise distribution from which data is drawn. Instead, a finite sample must be used as a surrogate when reasoning about random variables. In our formalism we wrote $X \xleftarrow{\$} \mathcal{P}$ to designate sampling of a value from a population. Given a dataset surrogate $\mathcal{D}$, this operation is implemented as an enumeration $\mathbf{x} \in \mathcal{D}$, with each element having probability $1/|\mathcal{D}|$. We will overload the notation and use $\mathcal{D}$ also as the random variable distributed in the manner just described. We assume here that the sensitive attribute $Z$ is a part of the random variable $X$.

The following sections use the following quantities to express complexity bounds, mostly overloading prior notations:

- $\mathcal{D}$ - The number of instances in the population dataset.
- $p$ - The number of expressions in a program $p$ being analyzed.
- $Z$ - The number of elements in the support of $Z$.
- $k$ - The maximum number of unique elements in support of every sub-expression, that is $\max_{p' \in p} ||$.
- $c$ - The number of decompositions in a given program. We will elaborate on this quantity under several circumstances later in this section.
- $b$ - The minimum branching factor of sub-expressions in a given program.

We will assume that the number of syntactic copies of any sub-expression in a pro-

gram is no more than some constant. This means we will ignore the asymptotic effect of decompositions with multiple copies of the same sub-program $p_1$.

The elementary operation in our algorithms is a lookup of a probability of a value according to some random variable. We pre-compute several probabilities related to reachability and contingency tables to aid in this operation. When we write "$p_1$ is reached", we mean that the evaluation of $p$, containing $p_1$, on a given instance $\mathbf{X}$, will reach the sub-expression $p_1$ (or that $p_1$ needs to be evaluated to evaluate $p$ on $\mathbf{X}$).

**Probability pre-computation**

For every decomposition $[\![p_2]\!](X, [\![p_1]\!]X) = [\![p]\!](X)$, we compute:

1. the r.v. $([\![p_1]\!]\mathbf{X}, \mathbf{X}_Z)$ for $\mathbf{X} \xleftarrow{\$} \mathcal{D}$,
2. the r.v. $\mathbf{X}|$ ($p_1$ is reached by $\mathbf{X}$) for $\mathbf{X} \xleftarrow{\$} \mathcal{D}$, and
3. the value $Pr_{\mathbf{X} \xleftarrow{\$} \mathcal{D}}$ ($p_1$ is reached by $\mathbf{X}$).

In point (1) above we write $\mathbf{X}_Z$ to designate the sensitive attribute component of $\mathbf{X}'$, hence this point computes the r.v. representing the output of $p_1$ along with the sensitive attribute $Z$. This will be used for the association computation.

The complexity of these probability computations varies depending on circumstances. In the worst case, the complexity is $\mathcal{O}(c\mathcal{D}p)$. However, under some assumptions related to programs $p$ and datasets $\mathcal{D}$, these bounds can be improved. We define two types of special cases which we call splitting and balanced:

**Definition 21.** *$p$ is* splitting *for $\mathcal{D}$ iff it has at most a constant number of reachable op operands (arguments of op expressions).*

The Decision trees are local for any dataset as they do not contain any op operands (they do contain relop operands). Further, if number of trees in random forests or number of coefficients in linear regression are held constant, then these models too are splitting for any dataset. The reasoning behind this definition is to prohibit arbitrarily large programs that do not split inputs using if-then-else expressions. It is possible to create such programs using arithmetic and boolean operations, but not using purely relational operations.

**Definition 22.** *$p$ is* $b$-balanced *for $\mathcal{D}$ iff all but a constant number of sub-expressions $e'$ have parent $e$ with $b > 1$ sub-expressions which split the instances that reach them approximately equally among their children.*

Balanced implies splitting as op operands do not satisfy the balanced split property hence there has to be only a constant number of them. Also, the definition is more general than necessary for the language presented in this paper where the branching factor is always 2 because the if-then-else expressions are the only ones that can satisfy the balanced split condition. Decision trees trained using sensible algorithms are usually balanced due to the branch split criteria employed preferring approximately equal splits of training instances. For the same reason, if the number of trees are held constant, then random forests are also likely to be balanced.

When $p$ is splitting for $\mathcal{D}$, the probability computation step reduces to $\mathcal{O}(\mathcal{D}p^2)$. This stems from the fact that the number of decompositions is asymptotically equal to the number of sub-expressions (limits to operands prevent more decompositions). Further, if $p$ is $b$-balanced for $\mathcal{D}$, the probability pre-computation reduces to $\mathcal{O}(\mathcal{D} \log_b \mathcal{D})$. In the

language presented $b = 2$. These bounds derive similarly to the typical divide and conquer program analysis; there are $\log_b \mathcal{D}$ layers of computation, each processing $\mathcal{D}$ instances.

### 6.7.2 Influence and Association

Our proxy definition further relies on two primary quantities used in Algorithm 2, influence and association. We describe the methods we use to compute them here.

**Quantitative decomposition influence** Given a decomposition $(p_1, u, p_2)$ of $p$, the influence of $p_1$ on $p_2$'s output is defined as:

$$\iota(p_1, p_2) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{X, X' \stackrel{\$}{\leftarrow} \mathcal{D}} \left[\Pr\left(\llbracket p_2 \rrbracket\left(\mathbf{X}, \llbracket p_1 \rrbracket \mathbf{X}\right) \neq \llbracket p_2 \rrbracket\left(\mathbf{X}, \llbracket p_1 \rrbracket \mathbf{X}'\right)\right)\right]$$

This quantity requires $\mathcal{D}^2$ samples to compute in general. Each sample takes at most $\mathcal{O}(p)$ time, for a total of $\mathcal{O}(p\mathcal{D}^2)$. However, we can take advantage of the pre-computations described in the prior section along with balanced reachability criteria and limited ranges of values in expression outputs to do better. We break down the definition of influence into two components based on reachability of $p_1$:

$$\begin{aligned}
\iota(p_1, p_2) &\stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{\mathbf{X}, \mathbf{X}' \stackrel{\$}{\leftarrow} \mathcal{D}} \left[\Pr\left(\llbracket p_2 \rrbracket\left(\mathbf{X}, \llbracket p_1 \rrbracket \mathbf{X}\right) \neq \llbracket p_2 \rrbracket\left(\mathbf{X}, \llbracket p_1 \rrbracket \mathbf{X}'\right)\right)\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{X} \stackrel{\$}{\leftarrow} \mathcal{D}} \left[\mathop{\mathbb{E}}_{\mathbf{X}' \stackrel{\$}{\leftarrow} \mathcal{D}} \left[\Pr\left(\llbracket p_2 \rrbracket\left(\mathbf{X}, \llbracket p_1 \rrbracket \mathbf{X}\right) \neq \llbracket p_2 \rrbracket\left(\mathbf{X}, \llbracket p_1 \rrbracket \mathbf{X}'\right)\right)\right]\right] \\
&= \Pr\left(p_1 \text{ not reached}\right) \cdot \mathop{\mathbb{E}}_{\mathbf{X} \stackrel{\$}{\leftarrow} \mathcal{D} | p_1 \text{ not reached}} [\cdots] \\
&\quad + \Pr\left(p_1 \text{ reached}\right) \cdot \mathop{\mathbb{E}}_{\mathbf{X} \stackrel{\$}{\leftarrow} \mathcal{D} | p_1 \text{ reached}} [\cdots] \\
&= 0 + \Pr\left(p_1 \text{ reached}\right) \cdot \\
&\quad \mathop{\mathbb{E}}_{\mathbf{X} \stackrel{\$}{\leftarrow} \mathcal{D} | p_1 \text{ reached}} \left[\mathop{\mathbb{E}}_{\mathbf{X} \stackrel{\$}{\leftarrow} \mathcal{D}} \left[\Pr\left(\llbracket p \rrbracket(\mathbf{X}) \neq \llbracket p_2 \rrbracket\left(\mathbf{X}, \llbracket p_1 \rrbracket \mathbf{X}'\right)\right)\right]\right] \\
&= \Pr\left(p_1 \text{ reached}\right) \cdot \\
&\quad \mathop{\mathbb{E}}_{\mathbf{X} \stackrel{\$}{\leftarrow} \mathcal{D} | p_1 \text{ reached}} \left[\mathop{\mathbb{E}}_{\mathbf{Y} \stackrel{\$}{\leftarrow} \llbracket p_1 \rrbracket \mathcal{D}} \left[\Pr\left(\llbracket p \rrbracket(\mathbf{X}) \neq \llbracket p_2 \rrbracket\left(\mathbf{X}, \mathbf{Y}\right)\right)\right]\right]
\end{aligned}$$

Note that all both random variables and one probability value in the final form of influence above have been pre-computed. Further, if the number of elements in the support of $\llbracket p_1 \rrbracket X$ is bounded by $k$, we compute influence using $k\mathcal{D}$ samples (at most $\mathcal{D}$ for $\mathbf{X}$ and at most $k$ for $\mathbf{Y}$), for total time of $\mathcal{O}(kp\mathcal{D})$.

Influence can also be estimated, $\hat{\iota}$ by taking a *sample* from $\mathcal{D} \times \mathcal{D}$. By Hoeffding's inequality [64], we select the subsample size $n$ to be at least $\log(2/\beta)/2\alpha^2$ to ensure that the probability of the error $\hat{\iota}(p_1, p_2) - \iota(p_1, p_2)$ being greater than $\beta$ is bounded by $\alpha$.

**Association** As discussed in Section 6.3, we use mutual information to measure the association between the output of a subprogram and $Z$. In our pre-computation steps we have already constructed the r.v. $(\llbracket p_1 \rrbracket \mathbf{X}, \mathbf{X}_Z)$ for $\mathbf{X} \stackrel{\$}{\leftarrow} \mathcal{D}$. This joint r.v. contains

both the subprogram outputs and the sensitive attribute hence it is sufficient to compute association metrics. In case of normalized mutual information, this can be done in time $\mathcal{O}(kZ)$, linear in the size of the support of this random variable.

### 6.7.3 Decompositions

The number of decompositions of a model determines the number of proxies that need to be checked in detection and repair algorithms. We consider two cases, splitting and non-splitting programs. For splitting models, the number of decompositions is bounded by the size of the program analyzed, whereas in case of non-splitting models, the number of decompositions can be exponential in the size of the model. These quantities are summarized in Table 6.2.

| | splitting | non-splitting |
|---|---|---|
| worst-case general | $\mathcal{O}(p)$ | $\mathcal{O}(2^p)$ |
| linear model with (constant or $f$ ) number of coefficients | $\mathcal{O}(1)$ | $\mathcal{O}\left(2^f\right)$ |
| decision tree of height $h$ | $\mathcal{O}\left(2^h\right)$ | $\mathcal{O}\left(2^h\right)$ |
| random forest of (constant or $t$) number of trees of height $h$ | $\mathcal{O}\left(2^h\right)$ | $\mathcal{O}\left(2^t 2^h\right)$ |

Table 6.2: The number of decompositions in various types of models. When we write "(constant or f)" we denote two cases: one in which a particular quantity is considered constant in a model making it satisfy the splitting condition, and one in which that same quantity is not held constant, falsifying the splitting condition.

### 6.7.4 Detection

The detection algorithm can be written $\mathcal{O}(A + B \cdot C)$, a combination of three components. $A$ is probability pre-computation as described earlier in this section, $B$ is the complexity of association and influence computations, and $C$ is the number of decompositions.

The complexity in terms of the number of decompositions under various conditions is summarized in Table 6.3. Instantiating the parameters, the overall complexity ranges from $\mathcal{O}(\mathcal{D} \log_b \mathcal{D} + p^2\mathcal{D})$ in case of models like balanced decision trees with a constant number of classes, to $\mathcal{O}(p2^p\mathcal{D}^2)$ in models with many values and associative expressions like linear regression. If the model size is held constant, these run-times become $\mathcal{O}(\mathcal{D} \log_b \mathcal{D})$ and $\mathcal{O}(\mathcal{D}^2)$, respectively.

| | |
|---|---|
| non-splitting | $\mathcal{O}\left(pc\left(\mathcal{D}+k\mathcal{D}\right)\right)$ |
| splitting | $\mathcal{O}\left(p\left(\mathcal{D}p+ck\mathcal{D}\right)\right)$ |
| $b-$balanced | $\mathcal{O}\left(\mathcal{D}\log_b \mathcal{D}+ckp\mathcal{D}\right)$ |

Table 6.3: The complexity of the detection algorithm under various conditions, as a function of the number of decompositions.

## 6.8 Related Work

### 6.8.1 Definition

**Minimizing disclosures** In the computer science literature, privacy has been thought of as the ability to protect against undesired flows of information to an adversary. Much of the machinery developed in cryptography, such as encryption, anonymous communication, private computation, and database privacy have been motivated by such a goal. Differential privacy [44] is one of the main pillars of privacy research in the case of computations over data aggregated from a number of individuals, where any information gained by an adversary observing the computation is not caused by an individual's participation. However, none of these technologies cover the important setting of individual-level data analytics, where one may want to share some information while hiding others from adversaries with arbitrary background knowledge. This absence is with good reason, as in the general case it is impossible to prevent flows of knowledge from individual-level data, while preserving the utility of such data, in the presence of arbitrary inferences that may leverage the background knowledge of an adversary [43]. In this work, we do not attempt to solve this problem either.

Nevertheless, the setting of individual level data analytics is pervasive, especially in the case of predictive systems that use machine learning. Since these systems are largely opaque, even developers do not have a handle on information they may be inadvertently using via inferences. Therefore, in this work, we make the case for proxy use restrictions in data driven systems and develop techniques to detect and repair violations of proxy use. Restrictions on information use, however do not supplant the need for other privacy enhancing technologies geared for restricting information collection and disclosure, which may be useful in conjunction with the enforcement of use restrictions. For example, when machine learning models are trained using personal data, it is desirable to minimize disclosures pertaining to individuals in the training set, and to reduce the use of protected information types for the individuals the models are applied to.

**Identifying explicit use** The privacy literature on use restrictions has typically focused on explicit use of protected information types, not on proxy use (see Tschantz et al. [136] for a survey and Lipton and Regan [87]). Recent work on discovering personal data use by black-box web services focuses mostly on explicit use of protected information types by

examining causal effects [32, 80]; some of this work also examines associational effects [79, 80]. Associational effects capture some forms of proxy use but not others as we argued in Section 6.3.

### 6.8.2 Detection and Repair Models

Our detection algorithm operates with white-box access to the prediction model. Prior work requires weaker access assumptions.

**Access to observational data** Detection techniques working under an associative use definition [50, 135] usually only require access to observational data about the behavior of the system.

**Access to black-box experimental data** Detection techniques working under an explicit use definition of information use [32, 80] typically require experimental access to the system. This access allows the analyst to control some inputs to the system and observe relevant outcomes.

The stronger white-box access level allows us to decompose the model and trace an intermediate computation that is a proxy. Such traceability is not afforded by the weaker access assumptions in prior work. Thus, we explore a different point in the space by giving up on the weaker access requirement to gain the ability to trace and repair proxy use.

Tramèr et al. [135] solve an important orthogonal problem of efficiently identifying populations where associations may appear. Since our definition is parametric in the choice of the population, their technique could allow identifying relevant populations for further analysis using our methods.

**Repair** Removal of violations of privacy can occur at different points of the typical machine learning pipeline. Adjusting the training dataset is the most popular approach, including variations that relabel only the class attribute [88], modify entire instances while maintaining the original schema [50], and transform the dataset into another space of features [45, 149]. Modifications to the training algorithm are specific to the trainer employed (or to a class of trainers). Adjustments to Naive Bayes [22] and trainers amiable to regularization [70] are examples. Several techniques for producing differentially-private machine learning models modify trained models by perturbing coefficients [14, 25]. Other differentially-private data analysis techniques [44] instead perturb the output by adding symmetric noise to the true results of statistical queries. All these repair techniques aim to minimize associations or inference from the outcomes rather than constrain use.

## 6.9 Discussion

**Beyond strict decomposition** Theorem 10 shows that a definition satisfying natural semantic properties is impossible. This result motivates our syntactic definition, parameterized by a programming language and a choice of program decomposition. In our im-

plementation, the choice of program decomposition is strict. It only considers single terms in its decomposition. However, proxies may be distributed across different terms in the program. As discussed in Section 6.4.1, single term decompositions can also deal with a restricted class of such distributed proxies. Our implementation does not identify situations where each of a large number of syntactically different proxies have weak influence but together combine to result in high influence. A stronger notion of program decomposition that allows a collection of multiple terms to be considered a proxy would identify such a case of proxy use.

The choice of program decomposition also has consequences for the tractability of the detection and repair algorithms. The detection and repair algorithms presented in this chapter currently enumerate through all possible subprograms in the worst case. Depending on the flexibility of the language chosen and the model[2] being expressed there could be an exponentially large number of subprograms, and our enumeration would be intractable.

Important directions of future work are therefore organized along two thrusts. The first thrust is to develop more flexible notions of program decompositions that identify a wide class of proxy uses for other kinds of machine learning models, including deep learning models that will likely require new kinds of abstraction techniques due to their large size. The second thrust is to identify scalable algorithms for detecting and repairing proxy use for these flexible notions of program decompositions.

**Data and access requirements** Our definitions and algorithms require (i) a specification of which attributes are protected, (ii) entail reasoning using data about these protected information types for individuals, and (iii) white box access to models and a representative dataset of inputs. Obtaining a complete specification of protected information types can be challenging when legal requirements and privacy expectations are vague regarding protected information types. However, in many cases, protected types are specified in laws and regulations governing the system under study (e.g., HIPAA, GDPR), and also stated in the data processor's privacy policies.

Further, data about protected information types is often not explicitly collected. Pregnancy status, for example, would rarely find itself as an explicit feature in a purchases database (though it was the case in the Target case). Therefore, to discover unwanted proxy uses of protected information types, an auditor might need to first infer the protected attribute from the collected data to the best extent available to them. Though it may seem ethically ambiguous to perform a protected inference in order to (discover and) prevent protected inferences, it is consistent with the view that privacy is a function of both information and the purpose for which that information is being used [136][3]. In our case, the inference and use of protected information by an auditor has a different (and ethically justified) purpose than potential inferences in model being audited. Further, protected information has already been used by public and private entities in pursuit of social good: affirmative action requires the inference or explicit recording of minority

[2]Though deep learning models can be expressed in the example language presented in this chapter, doing so would result in prohibitively large programs.

[3]This principle is exemplified by law in various jurisdictions including the PIPEDA Act in Canada [104], and the HIPAA Privacy Rule in the USA [101].

membership, search engines need to infer suicide tendency in order to show suicide prevention information in their search results[125], health conditions can potentially be detected early from search logs of affected individuals [102]. Supported by law and perception of public good, we think it justified to expect system owners be cooperative in providing the necessary information or aiding in the necessary inference for auditing.

Finally, in order to mitigate concerns over intellectual property due to access requirements for data and models, the analyst will need to be an internal auditor or trusted third party; existing privacy-compliance audits (Sen et al. [117]) that operate under similar requirements could be augmented with our methods.

**Normative judgments** Appropriateness decisions by the analyst will be made in accordance with legal requirements and ethical norms. Operationally, this task might fall on privacy compliance teams. In large companies, such teams include law, ethics, and technology experts. Our work exposes the specific points where these complex decisions need to be made. In our evaluation, we observed largely human-interpretable witnesses for proxies. For more complex models, additional methods from interpretable machine learning might be necessary to make witnesses understandable.

Another normative judgment is the choice of acceptable $\epsilon, \delta$ parameters. Similar to differential privacy, the choice of parameters requires identifying an appropriate balance between utility and privacy. Our quantitative theory could provide guidance to the oracle on how to prioritize efforts, e.g., by focusing on potentially blatant violations (high $\epsilon, \delta$ values).

## 6.10 Conclusion

We develop a theory of use privacy and proxy non-discrimination in data-driven systems. Distinctively, our approach constrains not only the direct use of protected information types but also their proxies (i.e. strong predictors), unless allowed by exceptions justified by ethical considerations.

We formalize proxy use and present a program analysis technique for detecting it in a model. In contrast to prior work, our analysis is white-box. The additional level of access enables our detection algorithm to provide a witness that localizes the use to a part of the algorithm. Recognizing that not all instances of proxy use of a protected information type are inappropriate, our theory of use privacy makes use of a normative judgment oracle that makes this appropriateness determination for a given witness. If the proxy use is deemed inappropriate, our repair algorithm uses the witness to transform the model into one that does not exhibit proxy use. Using a corpus of social datasets, our evaluation shows that these algorithms are able to detect proxy use instances that would be difficult to find using existing techniques, and subsequently remove them while maintaining acceptable classification performance.

# Chapter 7

# Conclusion and Future Work

In this dissertation, we described a basis for developing explanations for information use in data driven systems. These explanations are a means to enhancing trust in the predictions of machine learning models, and surfacing violations of privacy and fairness. Our approach combines two key elements: causal influence and associative interpretation. While these elements have been explored individually in prior work, combining the two opens up a rich space of techniques that we begin to explore in this dissertation. In Figure 7.1, we describe this space in terms of the factors under study, and the systems these influences are computed in. In this thesis, we focus on the part of the space that examines the influence of input factors and internal factors in trained models.

Further, each result described in this paper explains why a certain behavior occurs. In cases where the behavior is undesirable, we can attempt to leverage the explanation in order to repair the behavior. In the restricted cases of Chapter 3 and 6, we already explore this idea of leveraging explanations to aid the repair of models. Pursuing a general approach to repairing systems is another important line of future work.

## 7.1 Factors

For both proxy use (Chapter 6) and influence-directed explanations (Chapter 5), we consider the influence of single internal factors in isolation. However, viewing the influence of computations together might reveal interactions between these factors that we currently miss. An interesting future line of work would be to analyze the influence of a set of factors. For example, in a convolutional neural network the concept of the roof of a sports car might be represented by a certain combination of neuron values across different layers. Similarly, for a random forest different parts of different trees might combine to define a strong proxy for race than any single component. The key challenge in analyzing the influence of a set of factors is that an exponential number of sets could be potentially explored, and we need richer notions of program decomposition to address this challenge.

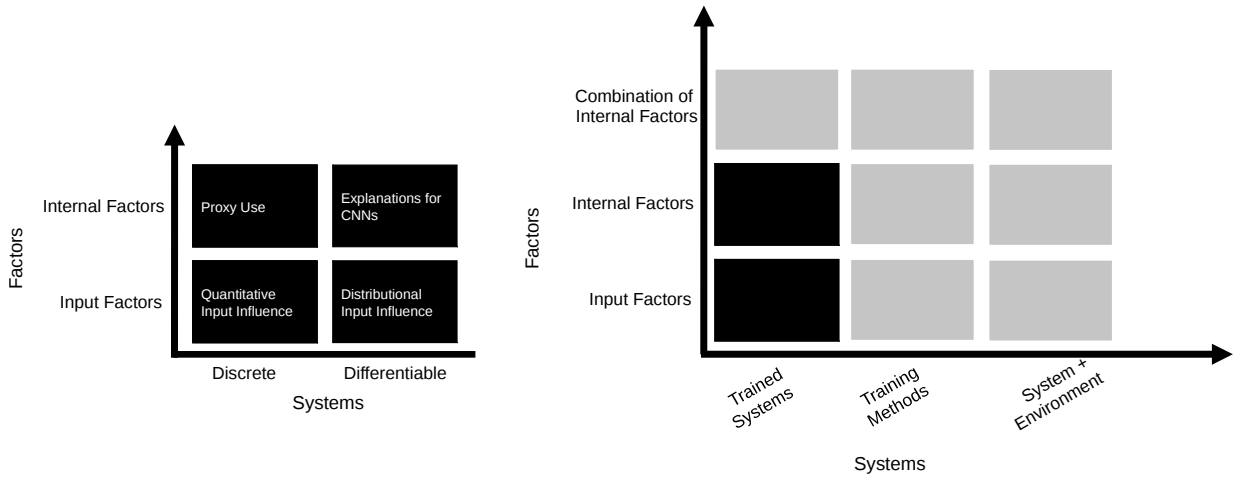



Figure 7.1: A summary of future work in influence-directed explanations described in terms of the systems we measure the influence in, and the factors we compute the influence of. In this dissertation we focus on the part of the space the looks at the influence of input and internal factors in trained models. These trained models are generated by complex training algorithms and operate in a rich environment. Explaining the behavior of these models in this expanded context is an important direction of future work. Further, generalizing to combinations of internal factors will allow us to capture concepts with a distributed internal representation.

## 7.2 Systems

In this dissertation, we focus on understanding how trained machine learning models use information. As the behavior of these models is directly governed by the training data, understanding the influence of parts of the training data on the behavior will enable repairs during the training process. Recent work by Chakarov et al. [23] and Koh et al. [77] study the use of influence functions in understanding how training data points affect the predictions of machine learning classifiers. An important direction of future work will extend these ideas from influence on predictions to influence on behaviors such as discrimination and privacy violations.

Further, analyzing causality in programs is relatively simple, but the true implications of the use of programs is borne out by the composition of the programs with the environment they operate in. For example, in predictive policing, policing certain neighborhoods might reduce criminal activity as criminals move to other neighborhoods, thus changing the underlying distribution the models were trained on. Similarly, providing loans to a certain community over a few years might improve the economic characteristics of that community and thus alter their creditworthiness. Understanding such effects of how algorithms lead to downstream effects on society will require combining an understanding of causal effects in algorithms with causal effects in society. While causal experimentation in programs is relatively simple, causal experimentation in society is expensive and time-consuming. Economists and social scientists have decades of research approximating causal experime-

nation from observational data. Combining the power of rapid causal experimentation in programs with techniques for causal inference in society will be essential in explaining the societal outcomes of the use of machine learning models.

## 7.3 Repair

Both counterfactual active learning (Chapter 3) and repair techniques for proxy use (Chapter 6) are techniques that leverage insights from explanations to repair the model. A more general approach to repair is another interesting line of future work. For example, once we identify that for CNNs a misclassification occurs because some set of incorrect neurons were fired, it should be possible to train those incorrect neurons on instances that correct the behavior of those neurons. This kind of retraining can be viewed as a form of counterfactual active learning but on the internal layers of a neural network.

# Appendix A

# Details for Proxy Use

## A.1 Algorithm for Detection

In this section we provide technical details about the detection algorithm skipped from the main body of the dissertation. In particular, we formally define the decomposition used in the implementation, how machine learning models are translated to the term language, and how associational tests mitigate spurious results due to sampling.

### A.1.1 Decomposition

Before we present the formal algorithm for detection, we need to develop notation for precisely denoting decompositions. Decomposition follows naturally from the subterm relation on expressions. However, as identical subterms can occur multiple times in an expression, care must be taken during substitution to distinguish between occurrences. For this reason we define substitution positionally, where the subterm of expression $e = \text{op}(e_1, \ldots, e_n)$ at position $q$, written $e|_q$, is defined inductively:

$$\text{op}(e_1, \ldots, e_n)|_q = \begin{cases} \text{op}(e_1, \ldots, e_n) & \text{if } q = \epsilon \\ e_i|_{q'} & \text{if } q = iq' \wedge 1 \leq i \leq n \\ \text{op}(e_{i_1}, \ldots, e_{i_k}) & \text{if } q = \{i_1, \ldots, i_k\} \\ \bot & \text{otherwise} \end{cases}$$

We denote $q$ as 'positional indicator'. Specifically, $q$ has the syntax of the following.
$\langle q \rangle ::= \epsilon \mid i\langle q \rangle \mid \{i_1, \ldots, i_k\}$
We then define the term obtained by substituting $s$ in $e$ at position $q$, written $e[s]_q$, to be the term where $e[s]_q|_q = s$, and $e[s]_q|_{q'} = e_{q'}$ for all $q'$ that are not prefixed by $q$. For a sequence of positions $q_1, \ldots, q_n$ and terms $s_1, \ldots, s_n$, we write $e[s_1, \ldots, s_n]_{q_1,\ldots,q_n}$ to denote the sequential replacement obtained in order from 1 to $n$. Given a program $p = \lambda \mathbf{x}.e$, we will often write $p|_q$ or $p[s]_q$ for brevity to refer to $e|_q$ and $e[s]_q$, respectively. The set of decompositions of a program $p$ is then defined by the set of positions $q$ such that $p|_q \neq \bot$. Given position $q$, the corresponding decomposition is simply $(\lambda \mathbf{x}.p|_q, u, \lambda \mathbf{x}, u.p[u]_q)$.

**Algorithm 5** Detection for expression programs.

**Require:** association ($d$), influence($\iota$) measures
  **procedure** ProxyDetect($p, \mathbf{X}, Z, \epsilon, \delta$)
    $P \leftarrow \varnothing$
    **for** each term $e$ appearing in $p$ **do**
      $p_1 \leftarrow \lambda x_1, \ldots, x_n.e$
      $Q \leftarrow \{q \mid p|_q = e\}$
      **for** each $k \in [1, \ldots, |Q|], (q_1, \ldots, q_k) \in Q$ **do**
        $p_2 \leftarrow \lambda x_1, \ldots, x_n, u.p[u]_{q_1,\ldots,q_k}$
        **if** $\iota(p_1, p_2) \geq \delta \wedge d([\![p_1]\!](\mathbf{X}), Z) \geq \epsilon$ **then**
          $P \leftarrow P \cup \{(p_1, p_2)\}$
        **end if**
      **end for**
    **end for**
    **return** $P$
  **end procedure**

**Example 7.** *Consider a simple model,*

$$\begin{aligned} p &= \lambda x, y.\mathbf{ite}(x + y \leq 0, 1, 0) \\ &= \lambda x, y.\mathbf{ite}(\leq (+(x, y), 0), 1, 0) \end{aligned}$$

*There are eight positions in the body expression, namely* $\{\epsilon, 1, 2, 3, 11, 12, 111, 112\}$*. The subexpression at position* 112 *is* $y$*, and* $p[u]_{11} = \mathbf{ite}(u \leq 0, 1, 0)$*. This corresponds to the decomposition:*

$$(\lambda x, y.x + y, u, \lambda x, y, u.\mathbf{ite}(u \leq 0, 1, 0))$$

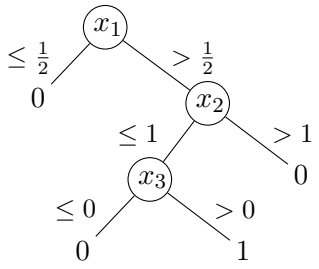
With this notation in place, we can formally describe the detection algorithm in Algorithm 5.

## A.1.2 Translation

This section describes the translation of machine learning models used in our implementation to the term language.

### Decision trees and Rule lists

Decision trees can be written in this language as nested **ite** terms, as shown in Figure A.1. The Boolean expression in each term corresponds to a guard, and the arithmetic expressions to either a proper subtree or a leaf. Bayesian rule lists are a special kinds of decision trees, where the left subtree is always a leaf.




$$\begin{aligned}&\textbf{ite}(x_1 \leq \tfrac{1}{2},\\ &\quad 0\\ &\quad \textbf{ite}(x_2 \leq 1,\\ &\qquad \textbf{ite}(x_3 \leq 0, 0, 1),\\ &\qquad 0)))\end{aligned}$$

Figure A.1: Decision tree and corresponding expression program.

**Linear models**

Linear regression models are expressed by direct translation into an arithmetic term, and linear classification models (e.g., logistic regression, linear support vector machines, Naive Bayes) are expressed as a single **ite** term, i.e.,

$$\text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \text{ becomes } \lambda \mathbf{x}.\textbf{ite}(\mathbf{w} \cdot \mathbf{x} + b \geq 0, 1, 0)$$

Importantly, the language supports $n$-ary operations when they are associative, and allows for rearranging operands according to associative and distributive equivalences. In other words, the language computes on terms modulo an equational theory. Without allowing such rearrangement, when a linear model is expressed using binary operators, such as $((((w_1 \times x_1) + (w_2 \times x_2)) + (w_3 \times x_3))$, then the algorithm cannot select the decomposition:

$$\begin{aligned}p_1 &= \lambda \mathbf{x}.(w_1 \times x_1) + (w_3 \times x_3)\\ p_2 &= \lambda \mathbf{x}, u.u + (w_2 \times x_2)\end{aligned}$$

**Decision Forests**

Decision forests are linear models where each linear term is a decision tree. We combine the two translations described above to obtain the term language representation for decision forests.

### A.1.3 Validity Testing

We use mutual information to determine the strength of the statistical association between $[\![p_1]\!](\mathbf{X})$ and $Z$. Each test of this metric against the threshold $\epsilon$ amounts to a hypothesis test against a null hypothesis which assumes that $d([\![p_1]\!](\mathbf{X}), Z) < \epsilon$. Because we potentially take this measure for each valid decomposition of $p$, it amounts to many simultaneous hypothesis tests from the same data source. To manage the likelihood of encountering false positives, we employ commonly-used statistical techniques. The first approach that we use is cross-validation. We partition the primary dataset $n$ times into training and validation sets, run Algorithm 5 on each training set, and confirm the reported proxy uses on the corresponding validation set. We only accept reported uses that appear at least $t$ times as valid.

**Algorithm 6** Witness-driven repair.

**Require:** association ($d$), influence ($\iota$), utility ($v$) measures, oracle ($\mathcal{O}$)
  **procedure** REPAIR($p, \mathbf{X}, Z, \epsilon, \delta$)
    $P \leftarrow \{d \in \text{PROXYDETECT}(p, \mathbf{X}, Z, \epsilon, \delta) : \text{ not } \mathcal{O}(d)\}$
    **if** $P \neq \emptyset$ **then**
      $(p_1, p_2) \leftarrow$ element of $P$
      $p' \leftarrow \text{PROXYREPAIR}(p, (p_1, p_2), \mathbf{X}, Z, \epsilon, \delta)$
      **return** $\text{REPAIR}(p', \mathbf{X}, Z, \epsilon, \delta)$
    **else**
      **return** $p$
    **end if**
  **end procedure**

The second approach uses bootstrap testing to compute a $p$-value for each estimate $\hat{d}(p_1(\mathbf{X}), Z)$, and applying Bonferroni correction [42] to account for the number of simultaneous hypothesis tests. Specifically, the bootstrap test that we apply takes $n$ samples of $(\mathbf{X}, Z)$, $[(\hat{X}_i, \hat{Z}_i)]_{1 \leq i \leq n}$, and permutes each $\hat{X}_i, \hat{Z}_i$ to account for the null hypothesis that $\mathbf{X}$ and $Z$ are independent. We then estimate the $p$-value by computing:

$$p = \frac{1}{n} \sum_{1 \leq i \leq n} 1(d(\hat{X}_i, \hat{Z}_i) < d(\llbracket p_1 \rrbracket(\mathbf{X}), Z))$$

After correction, we can bound the false positive discovery rate by only accepting instances that yield $p \leq \alpha$, for sufficiently small $\alpha$. We note, however, that this approach is only correct when the association strength $\epsilon = 1$, as the null hypothesis in this test assumes that $\llbracket p_1 \rrbracket$ is independent of $Z$. To use this approach in general, we would need to sample $[(\hat{X}_i, \hat{Z}_i)]_{1 \leq i \leq n}$ under the assumption that $d(\hat{X}_i, \hat{Z}_i) \geq \epsilon$. We leave this detail to future work.

## A.2 Algorithms for Repair

We now provide a formal description of the repair algorithms informally described in the paper. Algorithm 6, and 7 correspond to 3, and 4 respectively.

### A.2.1 Optimal constant selection

As constant terms cannot be examples of $(\epsilon, \delta)$-Proxy Use, there is freedom in their selections as replacements for implicated sub-programs. In Algorithm 7 we pick the replacement that optimizes some measure of utility of the patched program. If the given program was constructed as a classifier, we define utility as the patched program's prediction accuracy on the data set using 0-1 loss. Similarly, if the program were a regression model, $v$ would correspond to mean-squared error.

**Algorithm 7** Local Repair.

**Require:** association ($d$), influence ($\iota$), utility ($v$) measures

**procedure** ProxyRepair($p, (p_1, p_2), \mathbf{X}, Z, \epsilon, \delta$)
  $R \leftarrow \{\}$
  **for** each decomp. $(p_1', p_2')$ w/ $p_1'$ local to $p_1$ in $p_2$ **do**
    $r^* \leftarrow \arg\max_r v([u/r]p_2')$
    $(p_1'', p_2'') \leftarrow (p_1, p_2)$ with $r^*$ substituted for $p_1'$
    **if** $\iota(p_1'', p_2'') \leq \delta \vee d([\![p_1'']\!](\mathbf{X}), Z) \leq \epsilon$ **then**
      $p^* \leftarrow [u/r^*]p_2'$
      $R \leftarrow R \cup \{p^*\}$
    **end if**
  **end for**
  **return** $\arg\max_{p^* \in R} v(p^*)$
**end procedure**

If the program computes a continuous convex function, as in the case of most commonly-used regression models, then off-the-shelf convex optimization procedures can be used in this step. However, because we do not place restrictions on the functions computed by programs submitted for repair, the objective function might not satisfy the conditions necessary for efficient optimization. In these cases, it might be necessary to develop a specialized procedure for the model class. Below we describe such a procedure for the case of decision trees.

**Decision trees** Decision trees are typically used for classification of instances into a small number of classes $C$. For these models, the only replacement constants that will provide reasonable accuracy are those that belong to $C$, so in the worst case, the selection procedure must only consider a small finite set of candidates. However, it is possible to calculate the optimal constant with a single pass through the dataset.

Given a decomposition $(p_1, p_2)$ of $p$, let $\phi$ be the weakest formula over $p$'s variables such that $\forall \mathbf{x}.p_1(\mathbf{x}) = p(\mathbf{x})$. $\phi$ corresponds to the conjoined conditions on the path in $p$ prefixing $p_1$. We can then define the objective function:

$$v(r) = \sum_{\mathbf{x} \in \mathbf{X}} 1(\phi(\mathbf{x}) \rightarrow \mathbf{x}_c = r)$$

This objective is minimized when $r$ matches the greatest number of class labels for samples that pass through $p_1$. This minimizes classification error over $\mathbf{X}$, and is easily computed by taking the class-label mode of training samples that satisfy $\phi$.

**Example 8.** *Consider the tree in Figure A.1, and assume that $x_1$ and $x_2$ are distributed according to $\mathcal{N}(\frac{1}{2}, 1)$, and $x_3 = x_1 + x_2$. For simplicity, assume that the class label for each instance is given exactly by the tree. Then given the decomposition:*

$$p_1 = \lambda \mathbf{x}.\mathbf{ite}(x_3 \leq 0, 0, 1)$$
$$p_2 = \lambda \mathbf{x}, u.\mathbf{ite}(x_1 \leq 1/2, 0, \mathbf{ite}(x_2 \leq 1, u, 0))$$

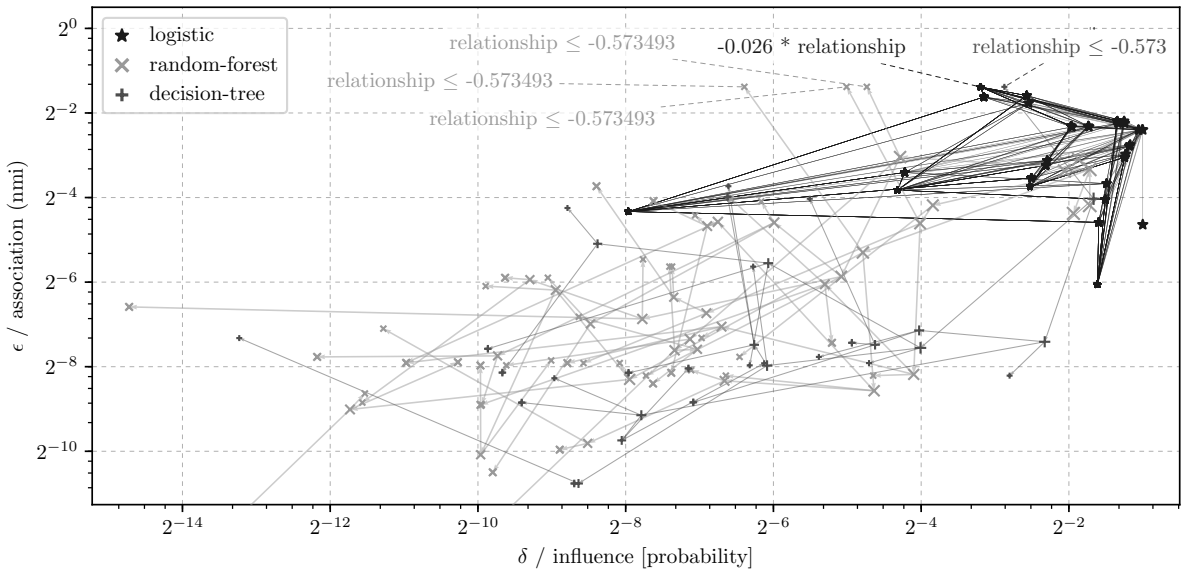



Figure A.2: The association with `marital_status` and influence of the sub-expressions of three similarly-sized models trained on the UCI Adult dataset (normalized): random forest ($\times$), decision tree ($+$), and logistic regression ($\star$). The arrows denote the sub-expression relationship among the expressions. Pointed out are several significant expressions that use the `relationship` feature. Note that in the random forest, the same associated expression appears in all three of the trees in that model.

*we need to find an optimal constant to replace the subtree rooted at* $x_3$*. In this case,* $\phi \stackrel{\text{def}}{=} x_1 > \frac{1}{2} \wedge x_2 \leq 1$*, so we select* $\mathbf{X}_\phi = \{\mathbf{x} \in \mathbf{X} | x_1 > \frac{1}{2} \wedge x_2 \leq 1\}$ *and take the mode of the empirical sample* $[p(\mathbf{x})]_{\mathbf{x} \in \mathbf{X}_\phi}$*.*

# Bibliography


[1] E.G. Griggs v. Duke Power Co., 401 U.S. 424, 91 S. Ct. 849, 28 L. Ed. 2d 158 (1977). 2

[2] National longitudinal surveys. http://www.bls.gov/nls/. 2, 2.6, 3.4.1

[3] Title VII of the civil rights act of 1964, 1964. URL https://www.eeoc.gov/laws/statutes/titlevii.cfm. Accessed Aug 13, 2016. 1.3, 6.2

[4] Equal Credit Opportunity Act (ECOA), 1974. URL https://www.justice.gov/crt/equal-credit-opportunity-act-3. Accessed Feb 24, 2017. 1, 1.1, 1.2.2, 1.3, 2, 2.6.6

[5] Indonesia - national contraceptive prevalence survey 1987, 2013. URL http://microdata.worldbank.org/index.php/catalog/1398/study-description. (Accessed Nov 11, 2016). 6.6.2

[6] AAO. American Academy of Ophthalmology: International clinical Diabetic Retinopathy disease severity scale. http://www.icoph.org, 2002. 5.2.3

[7] Philip Adler, Casey Falk, Sorelle A. Friedler, Tionney Nix, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. Auditing black-box models for indirect influence. *Knowledge and Information Systems*, 54(1):95–122, Jan 2018. ISSN 0219-3116. doi: 10.1007/s10115-017-1116-3. 2.8, 3, 3.1.2

[8] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and itâĂŹs biased against blacks. *ProPublica*, May 2016. URL https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. 1, 3, 6.2

[9] R. J. Aumann and L. S. Shapley. *Values of Non-Atomic Games*. Princeton University Press, 1974. 4.3

[10] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 07 2015. doi: 10.1371/journal.pone.0130140. 4.3, 5, 5.5, 5.5, 5.2

[11] Y. Bachrach, E. Markakis, E. Resnick, A.D. Procaccia, J.S. Rosenschein, and A. Saberi. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems*, 20(2):105–122, 2010. 2.4.1

[12] Paul Barford, Igor Canadi, Darja Krushevskaja, Qiang Ma, and S. Muthukrishnan.

Adscape: Harvesting and analyzing online display ads. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 597–608, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-2744-2. URL http://dx.doi.org/10.1145/2566486.2567992. 2.8

[13] S. Barocas and H. Nissenbaum. Big data's end run around procedural privacy protections. *Communications of the ACM*, 57(11):31–33, October 2014. 1, 2.7

[14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 464–473, 2014. 6.8.2

[15] Richard Berk and Justin Bleich. Forecasts of violence to inform sentencing decisions. *Journal of Quantitative Criminology*, 30(1):79–96, 2014. ISSN 1573-7799. 6.6

[16] Richard A. Berk, Susan B. Sorenson, and Geoffrey Barnes. Forecasting domestic violence: A machine learning approach to help inform arraignment decisions. *Journal of Empirical Legal Studies*, 13(1):94–115, 2016. ISSN 1740-1461. 6.6

[17] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2.6

[18] P. Bork, L.J. Jensen, C. von Mering, A.K. Ramani, I. Lee, and E.M. Marcott. Protein interaction networks from yeast to human. *Current Opinions in Structural Biology*, 14(3):292–299, 2004. 2.8

[19] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. 2.8, 3, 3.1.2, 6.6

[20] Warren Burger. Griggs v. duke power company. Opinion of the United States Supreme Court, March 1971. 1.2.2, 1

[21] Nanette Byrnes. An ai-fueled credit formula might help you get a loan. *ProPublica*, 2017. URL https://www.technologyreview.com/s/603604/an-ai-fueled-credit-formula-might-help-you-get-a-loan/. 3

[22] Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010. ISSN 1384-5810. 1.3, 2.7, 6.8.2

[23] Aleksandar Chakarov, Aditya V. Nori, Sriram K. Rajamani, Shayak Sen, and Deepak Vijaykeerthy. Debugging machine learning tasks. *CoRR*, abs/1603.07292, 2016. 7.2

[24] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Morgan and Claypool, 2011. 2.4.1

[25] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011. 6.8.2

[26] David Maxwell Chickering and David Heckerman. A decision theoretic approach to targeted advertising. In *Proceedings of the Sixteenth Conference on Uncertainty in*

*Artificial Intelligence*, UAI'00, pages 82–88, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9. 6.6

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20 (3):273–297, September 1995. ISSN 0885-6125. 6.6

[28] Paulo Cortez and Alice Maria Goncalves Silva. Using data mining to predict secondary school student performance. Technical Report, Department of Computer Science, University of Camerino, 2008. 6.6.2

[29] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012. 2.8, 6.3.4

[30] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–444, 1977. 1.3

[31] A. Datta, A. Datta, A.D. Procaccia, and Y. Zick. Influence in classification via cooperative game theory. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 511–517, 2015. 2.8

[32] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter Open, 2015. 1, 1.3, 1.3, 2.7, 2.8, 6.3.1, 6.3.1, 6.8.1, 6.8.2

[33] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings. *PoPETs*, 2015(1):92–112, 2015. 1.3

[34] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. 3, 3.1.2, 6.3.4

[35] Anupam Datta, Matthew Fredrikson, Gihyuk Ko, Piotr Mardziel, and Shayak Sen. Use privacy in data-driven systems: Theory and experiments with machine learnt programs. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1193–1210, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4946-8. 6

[36] Anupam Datta, Matthew Fredrikson, Gihyuk Ko, Piotr Mardziel, and Shayak Sen. Proxy non-discrimination in data-driven systems. Technical report, arXiv, July 2017. URL `http://arxiv.org/abs/1707.08120`. 3

[37] Wendy Davis. Ftc's julie brill tells ad tech companies to improve privacy protections, 2016. URL `http://www.mediapost.com/publications/article/259210/ftcs-julie-brill-tells-ad-tech-companies-to-impro.html`. Accessed Nov 11, 2016. 1.2.2

[38] Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Commun. ACM*, 20(7):504–513, 1977. 1.1, 1.3

[39] Pam Dixon and Robert Gellman. The scoring of america: How secret consumer scores threaten your privacy and your future, 2014. URL `http://www.worldprivacyforum.org/wp-content/uploads/2014/04/WPF_`

Scoring_of_America_April2014_fs.pdf. Accessed: 2016-11-05. 6.6.2

[40] Flávio du Pin Calmon and Nadia Fawaz. Privacy against statistical inference. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 1401–1408, October 2012. URL http://ieeexplore.ieee.org/abstract/document/6483382/. 1.3

[41] Charles Duhigg. How companies learn your secrets, 2012. URL http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html. (Accessed Aug 13, 2016). 1.2.2, 6

[42] Olive Jean Dunn. Estimation of the medians for dependent variables. 30(1):192–197, 03 1959. A.1.3

[43] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. ISBN 3-540-35907-9. URL http://dx.doi.org/10.1007/11787006_1. 1.2.2, 6.8.1

[44] Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006. 1.3, 2.5, 3, 6.8.1, 6.8.2

[45] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard 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. ISBN 978-1-4503-1115-1. URL http://doi.acm.org/10.1145/2090236.2090255. 1.2.2, 1.3, 2.7, 6.1, 6.8.2

[46] European Commission. General data protection regulation (GDPR). Regulation (EU) 2016/679, L119, May 2016. 1, 1.3, 3

[47] Executive Office of the President. Big data: A report on algorithmic systems, opportunity, and civil rights. Posted at https://www.whitehouse.gov/sites/default/files/microsites/ostp/2016_0504_data_discrimination.pdf, May 2016. Accessed Oct. 17, 2016. 1

[48] Federal Committee on Statistical Methodology. Statistical disclosure limitation methodology. Statistical Policy Working Paper 22, 2005. 1.3

[49] Federal Reserve. *Consumer Compliance Handbook*, chapter Federal Fair Lending Regulations and Statutes: Overview. Federal Reserve, 2016. 6.2

[50] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh 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. ISBN 978-1-4503-3664-2. URL http://doi.acm.org/10.1145/2783258.2783311. 1.2.2, 2.7, 6.3.1, 6.3.1, 6.8.2, 6.8.2

[51] Nicole Freeling. How big data is helping students graduate on time, 2016. URL https://www.universityofcalifornia.edu/news/how-big-data-helping-students-graduate-time. (Accessed Nov 11, 2016). 6.6.2

[52] Edward W. Frees, Richard A. Derrig, and Glenn Meyers. *Predictive Modeling Applications in Actuarial Science*. Cambridge University Press, 2014. 6.6

[53] Deepak Garg, Limin Jia, and Anupam Datta. Policy auditing over incomplete logs: theory, implementation and applications. In *Proceedings of The ACM Conference on Computer and Communications Security (CCS)*, 2011. 6.1

[54] Adrian Gepp, J. Holton Wilson, Kuldeep Kumar, and Sukanto Bhattacharya. A comparative analysis of decision trees vis-a-vis other computational data mining techniques in automotive insurance fraud detection. *Journal of Data Science*, 10 (3):537–561, 2012. 6.6

[55] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 580–587, 2014. ISBN 978-1-4799-5118-5. 5, 5.5

[56] Google. Privacy policy. Accessed Nov. 21, 2014. 6

[57] Saikat Guha, Bin Cheng, and Paul Francis. Challenges in measuring online advertising systems. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pages 81–87, New York, NY, USA, 2010. ISBN 978-1-4503-0483-2. URL http://doi.acm.org/10.1145/1879141.1879152. 2.8

[58] V Gulshan, L Peng, M Coram, and et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22):2402–2410, 2016. 5.2.3

[59] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944968. 2.8

[60] Robert Hare. *Manual For the Revised Psychopathy Checklist*. Multi-Health Systems, 2003. 6.6

[61] Benjamin Harold. The future of big data and analytics in k-12 education, 1 2016. 6.6.2

[62] Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2014)*. ACM, June 2014. URL http://research.microsoft.com/apps/pubs/default.aspx?id=226369. 1.3

[63] Yangbo He and Zhi Geng. Active learning of causal networks with intervention experiments and optimal designs. 9:2523–2547, 11 2008. 3

[64] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963. URL

http://www.jstor.org/stable/2282952. 2.4.2, 6.7.2

[65] Antti Hyttinen, Frederick Eberhardt, and Patrik O. Hoyer. Experiment selection for causal discovery. *Journal of Machine Learning Research*, 14:3041–3071, 2013. URL http://jmlr.org/papers/v14/hyttinen13a.html. 3

[66] Ideal Inc. How ai can stop unconscious bias in recruiting. https://ideal.com/unconscious-bias/, 2017. Accessed Nov. 22, 2017. 3

[67] D. Janzing, D. Balduzzi, M. Grosse-Wentrup, and B. Schölkopf. Quantifying causal influences. *Ann. Statist.*, 41(5):2324–2358, 10 2013. 2.8

[68] Zubin Jelveh and Michael Luca. Towards diagnosing accuracy loss in discrimination-aware classification: An application to predictive policing. *Fairness, Accountability and Transparency in Machine Learning*, 26(1):137–141, 2014. 2.6

[69] Kaggle. Diabetic Retinopathy Detection. https://www.kaggle.com/c/diabetic-retinopathy-detection, 2015. 5.2.3

[70] T. Kamishima, S. Akaho, and J. Sakuma. Fairness-aware learning through regularization approach. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW 2011)*, pages 643–650, 2011. 1.2.2, 1.3, 2.7, 6.8.2

[71] S.P. Kasiviswanathan, H.K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *Proceedings of the 49th IEEE Symposion on Foundations of Computer Science (FOCS 2008)*, pages 531–540, Oct 2008. 2, 2.5, 5

[72] A. Keinan, B. Sandbank, C.C. Hilgetag, I. Meilijson, and E. Ruppin. Fair attribution of functional contribution in artificial and biological networks. *Neural Computation*, 16(9):1887–1915, September 2004. 2.8

[73] Anthony Kennedy. Texas department of housing & community affairs v. the inclusive communities project, inc. Opinion of the United States Supreme Court, June 2015. 6.2

[74] Niki Kilbertus, Mateo Rojas-Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. *arXiv preprint arXiv:1706.02744*, 2017. 3

[75] Will Knight. The Dark Secret at the Heart of AI. *MIT Technology review*, Apr 2017. 5

[76] Gihyuk Ko, Piotr Mardziel, Shayak Sen, Anupam Datta, and Matt Fredrikson. Model checking white-box machine learning applications for fairness properties. Unpublished Manuscript. 1.2.2

[77] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. 70:1885–1894, 06–11 Aug 2017. 7.2

[78] Yann LeCun, LÃľon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998. 5.2, 5.2.1

[79] Mathias Lécuyer, Guillaume Ducoffe, Francis Lan, Andrei Papancea, Theofilos Pet-

sios, Riley Spahn, Augustin Chaintreau, and Roxana Geambasu. Xray: Enhancing the web's transparency with differential correlation. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 49–64, Berkeley, CA, USA, 2014. USENIX Association. ISBN 978-1-931971-15-7. 2.8, 6.8.1

[80] Mathias Lecuyer, Riley Spahn, Yannis Spiliopolous, Augustin Chaintreau, Roxana Geambasu, and Daniel Hsu. Sunlight: Fine-grained targeting detection at scale with statistical confidence. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 554–566, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. 2.8, 6.3.1, 6.3.1, 6.8.1, 6.8.2

[81] Lending Club. Lending club loan data. `https://www.kaggle.com/wendykan/lending-club-loan-data`, 2016. 2, 2.6, 3.4.1

[82] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Ann. Appl. Stat.*, 9(3):1350–1371, 09 2015. 2.8, 6.6

[83] Ninghui Li, Wahbeh H. Qardaji, and Dong Su. Provably private data anonymization: Or, k-anonymity meets differential privacy. *CoRR*, abs/1101.2604, 2011. 2.5, 5

[84] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`. 2, 2.3, 2.6

[85] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`. 3.4.1

[86] R. Lindelauf, H. Hamers, and B. Husslage. Cooperative game theoretic centrality analysis of terrorist networks: The cases of jemaah islamiyah and al qaeda. *European Journal of Operational Research*, 229(1):230–238, 2013. 2.8

[87] Richard J. Lipton and Kenneth W. Regan. Making public information secret, 2016. URL `https://rjlipton.wordpress.com/2016/05/20/making-public-information-secret/`. Accessed Aug 13, 2016. 1.2.2, 6.8.1

[88] Binh Thanh Luong, Salvatore Ruggieri, and Franco Turini. k-nn as an implementation of situation testing for discrimination discovery and prevention. 6.8.2

[89] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. *CoRR*, abs/1412.0035, 2014. URL `http://arxiv.org/abs/1412.0035`. 5, 5.5

[90] S. Maleki, L. Tran-Thanh, G. Hines, T. Rahwan, and A. Rogers. Bounding the estimation error of sampling-based shapley value approximation with/without stratifying. *CoRR*, abs/1306.4265, 2013. 2.4.1

[91] M. Maschler, E. Solan, and S. Zamir. *Game Theory*. Cambridge University Press, 2013. 2.3.1

[92] T.P. Michalak, T. Rahwan, P.L. Szczepanski, O. Skibski, R. Narayanam, M.J. Wooldridge, and N.R. Jennings. Computational analysis of connectivity games with applications to the investigation of terrorist networks. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 293–301,

2013. 2.8

[93] Microsoft. Microsoft privacy statement, September 2016. URL https://privacy.microsoft.com/en-us/privacystatement. 6

[94] Sumaria Mohan-Neill, Indira Neill Hoch, and Meng li. An analysis of us household socioeconomic profiles based on marital status and gender. *Journal of Economics and Economic Education Research*, (3), 9 2014. 6.6.1

[95] Andrew C. Myers and Barbara Liskov. Protecting privacy using the decentralized label model. *ACM Trans. Softw. Eng. Methodol.*, 9(4):410–442, 2000. 1.3

[96] Anh Mai Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *CoRR*, abs/1605.09304, 2016. URL http://arxiv.org/abs/1605.09304. 5, 5.5

[97] Helen Nissenbaum. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford Law Books. Stanford University Press, 2009. 6.1

[98] Northpointe. Practitioners guide to COMPAS. Technical report, Northpointe, Inc., August 2012. URL http://www.northpointeinc.com/files/technical_documents/FieldGuide2_081412.pdf. 1

[99] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107038324, 9781107038325. 2.3.1

[100] The President's Council of Advisors on Science and Technology. Big data and privacy: A technological perspective. Technical report, Executive Office of the President, May 2014. 1.2.2

[101] Office for Civil Rights. Summary of the HIPAA privacy rule. OCR Privacy Brief, U.S. Department of Health and Human Services, 2003. 6, 3

[102] John Paparrizos, Ryen W. White, and Eric Horvitz. Screening for pancreatic adenocarcinoma using signals from web search logs: Feasibility study and results. *Journal of Oncology Practice*, 12(8):737–744, 2016. doi: 10.1200/JOP.2015.010504. URL http://dx.doi.org/10.1200/JOP.2015.010504. PMID: 27271506. 6.9

[103] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, 2016. 5.2.1

[104] Parliament of Canada. Personal information protection and electronic documents act (PIPEDA). S.C. 2000, c. 5, 2000. 1, 1.3, 3

[105] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 560–568, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. 1.2.2

[106] J. Podesta, P. Pritzker, E.J. Moniz, J. Holdern, and J. Zients. Big data: Seizing opportunities, preserving values. Technical report, Executive Office of the President - the White House, May 2014. 2.7

[107] François Pottier and Vincent Simonet. Information flow inference for ml. In *POPL*, pages 319–330, 2002. 1.3

[108] Lewis F. Powell, Jr. Mcdonnell douglas corp. v. green. Opinion of the United States Supreme Court, May 1973. 6.2

[109] Harry Pratt, Frans Coenen, Deborah M. Broadbent, Simon P. Harding, and Yalin Zheng. Convolutional neural networks for diabetic retinopathy. *Procedia Computer Science*, 90:200 – 205, 2016. 5.2.3

[110] A. Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561, Berkeley, Calif., 1961. University of California Press. URL `http://projecteuclid.org/euclid.bsmsp/1200512181`. 2.8

[111] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939778. URL `http://doi.acm.org/10.1145/2939672.2939778`. 1, 3, 5, 5.5

[112] P. K. Rubenstein, I. Tolstikhin, P. Hennig, and B. Schölkopf. Probabilistic active learning of functions in structural causal models. 2017. 3

[113] Stefan Rüping. *Learning interpretable models.* PhD thesis, Dortmund University of Technology, 2006. http://d-nb.info/997491736. 2.8

[114] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. 1.2.1, 5, 5.3.2

[115] Salman Salamatian, Amy Zhang, Flávio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. Managing your private and public data: Bringing down inference attacks against your privacy. *J. Sel. Topics Signal Processing*, 9(7):1240–1255, 2015. URL `http://dx.doi.org/10.1109/JSTSP.2015.2442227`. 1.3

[116] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. R. MÃijller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, Nov 2017. 5.2.1

[117] Shayak Sen, Saikat Guha, Anupam Datta, Sriram K. Rajamani, Janice Tsai, and Jeannette M. Wing. Bootstrapping privacy compliance in big data systems. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 327–342, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-4686-0. URL `http://dx.doi.org/10.1109/SP.2014.28`. 6, 6.9

[118] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. ISSN 1538-7305. 2.8

[119] L. S. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *The American Political Science Review*, 48(3):787–792, 1954. 2.3.1

[120] L.S. Shapley. A value for $n$-person games. In *Contributions to the Theory of Games, vol. 2*, Annals of Mathematics Studies, no. 28, pages 307–317. Princeton University Press, 1953. 2.3, 2.3.1

[121] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1.2.1, 5, 5.3.1, 5.4, 5.5, 5.3.2

[122] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *ArXiv e-prints*, 2014. 4.3, 5, 5.5, 5.5, 5.2

[123] G. Smith. Quantifying information flow using min-entropy. In *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST 2011)*, pages 159–167, 2011. 1.3, 2.8

[124] Geoffrey Smith. Recent developments in quantitative information flow (invited tutorial). In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, LICS '15, pages 23–31, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4799-8875-4. URL `http://dx.doi.org/10.1109/LICS.2015.13`. 1.2.2

[125] S.E. Smith. How do search engines respond when you google âĂŸsuicideâĂŹ?, 2015. URL `https://www.dailydot.com/via/germanwings-suicide-hotline/`. Accessed May 15, 2017. 6.9

[126] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2nd edition, 2000. 3

[127] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *ICLR*, 2015. 4.3, 5, 5.5, 5.5

[128] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18, March 2010. ISSN 1532-4435. 2.8

[129] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *ArXiv e-prints*, 2017. 4.1, 4.3, 5, 5.1, 5.2.1, 5.4, 5.3.1, 5.3.2, 5.5, 5.5, 5.2

[130] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL `http://arxiv.org/abs/1409.4842`. 5.2.3

[131] Hugo Teufel III. Privacy policy guidance memorandum: The fair information practice principles: Framework for privacy policy at the Department of Homeland Security. Memorandum Number: 2008-01, December 2008. URL `https://www.dhs.`

gov/xlibrary/assets/privacy/privacy_policyguide_2008-01.pdf. 1, 1.3

[132] The National Center for Fair and Open Testing. 850+ colleges and universities that do not use sat/act scores to admit substantial numbers of students into bachelor degree programs, 2015. URL http://www.fairtest.org/university/optional. 2.7

[133] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society Series B*, 73(3):273–282, 2011. URL http://EconPapers.repec.org/RePEc:bla:jorssb:v:73:y:2011:i:3:p:273-282. 2.8

[134] Simon Tong and Daphne Koller. Active learning for structure in bayesian networks. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, pages 863–869, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 3

[135] Florian Tramèr, Vaggelis Atlidakis, Roxana Geambasu, Daniel J. Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. Discovering unwarranted associations in data-driven applications with the fairtest testing toolkit. *CoRR*, abs/1510.02377, 2015. URL http://arxiv.org/abs/1510.02377. 1.2.2, 6.3.1, 6.3.1, 6.3.4, 6.8.2, 6.8.2

[136] Michael Carl Tschantz, Anupam Datta, and Jeannette M. Wing. Formalizing and enforcing purpose restrictions in privacy policies. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 176–190, Washington, DC, USA, 2012. 1.3, 6, 6.8.1, 6.9

[137] Michael Carl Tschantz, Anupam Datta, and Jeannette M. Wing. Purpose restrictions on information use. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS)*, volume 8134 of *Lecture Notes in Computer Science*, pages 610–627. Springer Berlin Heidelberg, 2013. 6

[138] Joseph Turow. *The Daily You: How the New Advertising Industry Is Defining Your Identity and Your Worth*. Yale University Press, 2011. ISBN 9780300165012. 1.2.2

[139] Findings under the Personal Information Protection and Electronic Documents Act (PIPEDA). Use of sensitive health information for targeting of google ads raises privacy concerns, 2014. URL https://www.priv.gc.ca/cf-dc/2014/2014_001_0114_e.asp. Accessed Aug 13, 2016. 1.2.2, 1.2.2

[140] George Washington University. Standardized test scores will be optional for gw applicants, 2015. URL https://gwtoday.gwu.edu/standardized-test-scores-will-be-optional-gw-applicants. 2.7

[141] US Consumer Finance Protection Bureau. Supporting consumer-friendly innovation: Announcing our first no-action letter. News Release, September 2017. URL https://www.consumerfinance.gov/about-us/blog/supporting-consumer-friendly-innovation-announcing-our-first-no-action-letter/. 1

[142] U.S. Federal Goverment. Part 1607—uniform guidelines on employee selec-

tion procedures. Code of Federal Regulations, Title 29 - Labor, Vol. 4, 1978. URL https://www.gpo.gov/fdsys/pkg/CFR-2011-title29-vol4/xml/CFR-2011-title29-vol4-part1607.xml. 6.2

[143] US Food and Drug Administration. Fda permits marketing of artificial intelligence-based device to detect certain diabetes-related eye problems. News Release, April 2018. URL https://www.fda.gov/NewsEvents/Newsroom/PressAnnouncements/ucm604357.htm. 1

[144] Berk Ustun, Stefano TracÃă, and Cynthia Rudin. Supersparse linear integer models for interpretable classification. *ArXiv e-prints*, 2013. URL http://arxiv.org/pdf/1306.5860v1. 2.8

[145] Siva Viswanathan. Business intelligence and predictive analytics for financial services: The untapped potential of soft information. In *Digits: Center for Digital Innovation, Technology, and Strategy "Research in Practice" Paper Series*. Robert H. Smith School of Business, University of Maryland, 2010. 6.6.1

[146] H. P. Young. Individual contribution and just compensation. In Alvin E. Roth, editor, *The Shapley Value*, chapter 17, pages 267–278. Cambridge University Press, 1988. 4.3

[147] H.P. Young. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72, 1985. 2.3.2, 1

[148] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014. 4.3, 5, 5.5, 5.5, 5.2

[149] Richard S. Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. Learning fair representations. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 325–333. JMLR.org, 2013. URL http://jmlr.org/proceedings/papers/v28/zemel13.html. 1.2.2, 1.3, 2.7, 6.8.2