Minds and Machines (2023) 33:541–562
https://doi.org/10.1007/s11023-022-09591-0

ORIGINAL ARTICLE

Spotting When Algorithms Are Wrong

Stefan Buijsman[1] · Herman Veluwenkamp[1]

Received: 31 August 2021 / Accepted: 16 January 2022 / Published online: 21 January 2022

Abstract

Users of sociotechnical systems often have no way to independently verify whether the system output which they use to make decisions is correct; they are epistemically dependent on the system. We argue that this leads to problems when the system is wrong, namely to bad decisions and violations of the norm of practical reasoning. To prevent this from occurring we suggest the implementation of defeaters: information that a system is unreliable in a specific case (undercutting defeat) or independent information that the output is wrong (rebutting defeat). Practically, we suggest to design defeaters based on the different ways in which a system might produce erroneous outputs, and analyse this suggestion with a case study of the risk classification algorithm used by the Dutch tax agency.

Keywords Sociotechnical systems · Oversight · Defeaters · Epistemic dependence

1 Introduction

We make more and more decisions in the context of sociotechnical systems, having to reason with the information we receive from the system and act based on the options it presents us with. Desiere et al. (2019) offer a range of such examples in use by public employment services, and the use of COMPAS by the US judicial system and HireVue's AI system that automatically scores job applicants are two more examples where users end up relying on system output to make (high impact) decisions. This sociotechnical context[1] brings with it a conceptual challenge: how can we design the overarching systems such that their use leads to optimal decisions?

[1] One where humans, institutions and technical (possibly but not necessarily AI) elements interact to produce goal-directed behaviour. Sociotechnical systems are the overarching systems which encapsulate these interactions.

✉ Stefan Buijsman
s.n.r.buijsman@tudelft.nl

Herman Veluwenkamp
H.M.Veluwenkamp@tudelft.nl

[1] TU Delft, Jaffalaan 5, 2628 BX Delft, The Netherlands

One aspect of this is getting users to rely on the system when its outputs are correct. That is a challenge of its own, as users frequently do not follow the advice of algorithms as much as they should (e.g. Erlei et al., 2020; Logg et al., 2019). However, we want to look at the opposite challenge: how to design for the case where the algorithm is mistaken. What happens in these cases, and what are the different ways in which we might try to make users aware of system errors?

This is a pressing issue, as mistakes of this kind are bound to happen and there are a good number of historical examples to be found. The case of Stanislav Petrov, who averted nuclear war by choosing to report an incoming missile alert as a system malfunction to Soviet command (Chan, 2017), is a particularly admirable one. On September 26, 1983, their early-warning system gave off a loud alarm that five intercontinental ballistic missiles had been launched by the US. The system went from reporting 'Launch' to 'Missile strike', indicating that this information was of the "highest" level of certainty. Soviet ground radar had detected nothing, but would not have during the first few minutes of a missile launch. Still, Petrov decided after five minutes (of the 25 that he had before a missile might strike) that it was probably a false alarm, partly based on the idea that no first strike would have only five missiles, partly based on his distrust in the system. It turned out afterwards that the early-warning system was triggered by the reflection of sunlight off the tops of clouds. Though the automated system malfunctioned, a good final decision was reached thanks to the user. The sociotechnical system as a whole managed to avert a catastrophic mistake.

This is the ideal situation, contrasted by cases where a sociotechnical system, due to interactions between users and the automated parts, leads to bad decision making. One such example is Iran Air Flight 655, which was mistaken for a military plane and shot down by USS Vincennes shortly after take-off (Rochlin, 1991). Though the onboard systems did not malfunction, a good amount went wrong. The system correctly identified the plane, which had a flight path crossing over the ship, as civilian on the first ping. The second identification was as a military aircraft, though, but only because the operator forgot to reset the range of the system and picked up a stationary airplane on a nearby Iranian base instead of Flight 655. Meanwhile, the commercial flight log containing the departure information of Flight 655 was briefly consulted, but sharp turns of the ship sent the documents flying through the command center. As a result, the flight was missed, partly due to confusion about the four different time zones in the area by the operator checking the schedule. Further conflicting information, such as the fact that the plane was climbing, and not (as would be the case in an attack run on the Vincennes) descending, was tracked but only on screens far from where the flight path was tracked. Consequently, the operators missed it, and made the dramatic mistake of shooting down the plane, resulting in 290 deaths.

More modern systems, incorporating machine learning (ML) algorithms, run similar risks. HireVue's system presents recruiters with a score on how well a candidate did during the interview. And while recruiters are in principle capable of watching the recorded interview themselves, they are under too much time pressure to make that a realistic option. As a result they have to rely (in practice) on the information the system presents, in the form of a suitability score. Whether that score is always an accurate reflection of the qualities of the candidate is far from clear, and so hiring managers become dependent on a system that is opaque and whose mistakes will be hard to spot without taking the time to watch the recordings. Consequently, companies may end up making mistakes in the hiring process, just as the Dutch tax authority made serious mistakes with its algorithm to detect fraudsters in the case study worked out in Sect. 5.

Why exactly do sociotechnical systems give rise to such cases and what might one do to avert bad outcomes? That is the question we aim to tackle in this paper. We start with analyzing the underlying reason for the issue in Sect. 2, namely the epistemic dependence of users on the automated system. This hampers apt practical decision making in cases where the system gives wrong, or misleading, information. Then, in Sect. 3, we present what we think is the most promising way to prevent mistakes in these cases: providing defeaters. Defeaters are pieces of information, like the commercial flight log, that undermine the (faulty) conclusion of the system. We distinguish between two types of defeaters, undercutting and rebutting, and discuss how they fit into the theoretical picture from Sect. 2. Still, the ultimate goal is to enable sociotechnical design that allows users to make the best possible decision, relying on the information from the technical parts. So, in Sect. 4 we discuss how one might, in general, design for defeaters by looking at the different reasons for incorrect system output. We end by looking at a case study—regarding the fraud system used by the Dutch tax agency—to show how design for defeaters might work in practice, in Sect. 5.

2 Epistemic Dependence and practical decisions

2.1 Epistemic Dependence

The operators of the USS Vincennes and hiring managers using HireVue's system have to rely on the information of the system. The operators of the Vincennes only worked with the information of the system tracking flight routes and are to be blamed for their mistake primarily because conflicting information was readily available as part of the sociotechnical system. Hiring managers do have recordings available to check HireVue's algorithmic scores, but are often assigned too little time per decision to make this a viable option. Both cases in this way exemplify a general pattern. Users of sociotechnical systems are typically epistemically dependent on the system, as van den Hoven (1998) already discusses. This epistemic dependence creates problems when the system is wrong, as it is precisely because of users' dependence that they will have trouble differentiating correct system output from incorrect system output. First, however, we should be clearer on what epistemic dependence amounts to exactly. van den Hoven (1998) offers the following gloss: "A user B is epistemically dependent on system S if B cannot but cast her account of what she did and why she did it wholly in terms of the system. She may be unable to put forward 'system independent reasons'" (van den Hoven, 1998, p. 104) Similarly, Rooksby (2009), in a critique of the wider account offered by van den Hoven (1998), compares the case to dependence on experts, where: "One is epistemically dependent on an expert when one has good reason to believe true a claim held true by the expert, but cannot assess its truth oneself." (Rooksby, 2009, p. 82). She follows the account of Hardwig (1985) with this definition, and gives the same idea of being unable to verify the veracity of a claim independently of the system one uses, though note the addition that one needs to have good reason to believe the claim (i.e. trust the expert).

While we do not think that it is necessary to trust a system to be epistemically dependent on it (e.g. one may simply be given no other choice but to use the system despite its flaws), it is common that the system in use is generally reliable. When users are given a choice to dissent from the system, possibly presented with the input information, they can after all decide to ignore the system outputs and make decisions on their own. In fact, this happens in practice and trust calibration is a difficult issue for those working on Human–Computer Interaction (Erlei et al., 2020; Logg et al., 2019). As such, the most common scenario is likely one where users have the option to follow or not follow the system, and choose to do so based on a measure of trust in (or epistemic authority of) the system—though other factors such as time constraints for hiring managers can certainly contribute.

One reason why some underlying trust in the system is needed is that over time users might have the ability to estimate the overall reliability of the system, if they get some kind of feedback on their decisions. For the USS Vincennes there must have been a good measure of trust in the systems used for that reason: the system typically identified aircraft correctly and mistakes such as the one they made are fortunately extremely rare. Such feedback can be hard to come by, as e.g. in the case of hiring managers who do not see the good candidates that the system erroneously rejected, and might get only limited follow-up on the candidates they did let through to the next round. Moreover, this overall sense of reliability only reduces epistemic dependence in part, as it offers little help in deciding in individual cases whether the system is correct or not. Epistemic dependence will thus likely come hand in hand with trust in the system, possibly established over time, though we do not consider it absolutely necessary for such dependence to occur.

Finally, we should consider the even stronger definition of epistemic dependence put forward by Adam Carter (2017, 2021)—applicable here if one takes epistemic dependence to be the case where we do not have epistemic autonomy—and its relation to cognitive enhancement. His final formulation of the conditions for epistemic autonomy goes as follows:

> S's belief that p is epistemically autonomous (viz., autonomous [in a] way
> that is necessary for propositional knowledge) at a time, t, if and only
> if p has a compulsion-free history at t; and this is a history it has if
> and only if it's not the case that S came to acquire her belief that p in a way that: (i) bypasses or preempts S's cognitive competences, and (ii) the bypassing or preemption of such competences issues in S's being unable to shed P. (Adam Carter, 2021, p. 34)

This notion, as well as his idea in Adam Carter (2017) that accessibility, automatic endorsement and cognitive ownership are important factors in determining epistemic autonomy, suggest a more stringent concept than that envisioned by van den

Hoven (1998) and Rooksby (2009). He primarily discusses cases such as Truetemp (Lehrer, 1990), where a person—Mr. Truetemp—gets a small machine implanted into his brain that produces correct beliefs about the current temperature, without his knowledge. Such beliefs are not, on Adam Carter's definition, epistemically autonomous, as the machine bypasses Truetemp's cognitive competences. Now, sociotechnical systems will typically not be as extreme as in the Truetemp case, but this definition might still give us similar results to that of van den Hoven and Rooksby. Their guiding idea is that it will be difficult to shed the belief formed on the basis of the automated system. For example: if the system tells you that a candidate is unsuitable and has a record of high accuracy, it will be difficult to disagree (and defend the disagreement later on) if little other information is available. If the USS Vincennes systems warn of an incoming attack, it is hard not to believe this is the case when those systems are trusted. Depending on the transparency of the system, it might also be difficult to ascertain how this conclusion was reached, in effect bypassing our cognitive competences/reasoning. The extreme case is where an automated system simply tells you: P, but gives no supporting evidence for it and one has no alternative sources of information on whether P is true. In that case it seems that this definition of epistemic autonomy just as readily tells us that the user of such a system is epistemically dependent on it, though it is focussed on severe cases of epistemic dependence. Still, we do think that the broader definition, leaving out the clause on bypassing cognitive competences, is the better one.

For even if the system that the operators of the USS Vincennes worked with was more transparent, they would still have to rely entirely on the information the system provided. And while explainability methods (Guidotti et al., 2018 for a review) can help users spot when the algorithm is wrong in some instances, and as such constitute a way of providing information about the system that is independent from the algorithm itself (in Sect. 4 we e.g. discuss saliency maps as potential defeaters) they are not the only way to reduce epistemic dependence. Indeed, users receive more information about the system's decision, and might be better able to weigh the strength of the evidence that the system outputs P, but at best this helps assess the accuracy of the system that is explained. Explainability methods do count as independent verification here, as they are not part of the original system/algorithm. However, they are limited in scope, as they cannot provide the kind of additional evidence that e.g. the flight log does in the USS Vincennes example, or as the rebutting defeaters discussed in Sect. 5 do. Furthermore, explainability methods likely do not catch all the ways in which an algorithm might produce incorrect outputs (e.g. outlier detection is highly relevant but not a way to make the algorithm more transparent) and so transparency does not imply that one is no longer epistemically dependent on a system.

To further illustrate this difference, consider the definition of algorithmic opacity by Humphreys: an algorithm is opaque relative to a cognitive agent X at time t "just in case X does not know at t all of the epistemically relevant elements of the process" (Humphreys, 2009, p. 618). It is possible to know all the epistemically relevant elements of the process (i.e. fully understand how the system reaches its output) while still being dependent on the system, because of a lack of outside information. In an extreme case, if an algorithm flips a coin to decide between A and not-A, then I can fully understand the system but would still be in a situation of epistemic dependence if I am not given additional information and am forced to make a choice given only the system information. If we compare this to opaque ML systems, then their opacity makes it harder to spot when the algorithm is unreliable, but solving that opacity is not guaranteed to give users the ability to always spot the mistakes of the algorithm. That may need outside information as a way to verify the outputs. It is this lack of outside information, in combination with possible difficulties to assess the reliability of the system in specific instances, that leads to epistemic dependence.

So, we say that a user is epistemically dependent on a sociotechnical system S iff it is difficult for the user to independently verify the outputs of S. Of course, epistemic dependence comes in degrees, as the difficulty of independent verification can vary, and our cases so far have been examples with very strong epistemic dependence. We'll consider weaker cases in Sect. 4, but for all cases our claim is that this epistemic dependence on sociotechnical systems leads to difficulties in practical decision making for their users. In particular, epistemic dependence is problematic when the system is wrong, as its definition implies that it will be hard for a user of a sociotechnical system to correct the mistaken output. The result is, so we analyse these situations, that users will take the output of the automated system as direct input for their practical reasoning, and base their decisions (primarily) on that. In other words, we consider cases where users typically trust the system (and thus consider it an epistemic authority). Given the leeway that users are typically given in socio-technical systems and the fact that they certainly do not always rely on the system (as a matter of empirical fact), trust or the absence of alternatives are the two additional elements leading to the use of the system's output in practical reasoning. The issue then is that even systems that users trust are imperfect and will make mistakes, though users will not be in a position to recognize when that happens.

Our goal is to offer solutions to this issue, by designing sociotechnical systems in such a way that users are also presented with information that speaks against the main algorithm in use. This reduces epistemic dependence on the main algorithm, and hopefully prevents cases where it would lead the user into errors. However, it usually does not reduce epistemic dependence on the sociotechnical system as a whole, in which this additional information will be integrated. Before looking at solutions, however, we analyse the problem somewhat further. For incorrect information from the system can not only lead to wrong decisions, it can also conflict with the norms of practical reasoning. We turn to this point next.

2.2 Norms of Practical Reasoning

Users of sociotechnical systems will have to make decisions based on the information they receive. Ideally, we want to design systems such that good decisions are fostered and bad decisions are avoided. It is with this goal in mind that we consider the norms for practical reasoning to be relevant. For those decisions that the user makes as operator of the sociotechnical system, some practical reasoning takes place (even if it is as little as doing whatever the system suggests). The outputs of the system will act as inputs to this reasoning process, and if there is a case of epistemic dependence on the system then there will be relatively few additional inputs for the reasoning process. As a result, whether or not the actual reasoning meets the norms of practical reasoning will largely depend on the characteristics of the system output. With that in mind we first briefly look at the (sizable) literature on norms of practical reasoning to determine what conditions system output should meet.

Opinions on what this norm is differ, and we aim for our analysis to be independent of any particular choice here—it should work regardless of what exactly the norm of practical reasoning turns out to be. A fairly popular idea, however, is that there is a knowledge norm in place (Hawthorne & Stanley, 2008; Jackson, 2012; Mehta, 2016; Mueller, 2021; Williamson, 2005). Though there are variations across accounts, the basic idea is as follows: "Where one's choice is p-dependent, it is appropriate to treat the proposition that p as a reason for acting iff you know that p" (Hawthorne & Stanley, 2008, p. 578). In other words, one should only use what one knows as a basis for one's practical reasoning. That norm is violated when the user of a sociotechnical system relies on faulty system information (since knowledge is factive). So, in the case of a hiring manager receiving an incorrect suitability score he cannot know that the candidate is unsuitable, as the score is wrong. Similarly, for the USS Vincennes, the operator based his reasoning on the identification of an aircraft as 'military', but since the system picked up a different aircraft the belief that 'the plane approaching us at speed is a military aircraft' was not a piece of knowledge. Ultimately that mistaken belief, formed on the basis of the automated system, was the case for their decision to shoot. It is tempting, then, to analyse these cases as ones where the underlying problem was the violation of the knowledge norm of practical reasoning. That can happen in a far wider range of cases, but the epistemic dependence on the system makes it much harder to spot such norm violations.

The fact that it can happen in far more cases also points to an issue for the knowledge norm: there are plenty of violations of this norm where the person acting without knowledge is not to blame. If I base my decision to walk to the fridge on a belief that there is a carton of milk in it (because I bought one yesterday), but it turns out that the milk was stolen, then I have violated the knowledge norm of practical reasoning, but I still acted rationally and am blameless. Similarly, while the Vincennes operators may be to blame for missing readily available information showing their error, a hiring manager is not so clearly to blame as it was unrealistic (given the time constraints and stakes) to check the system output more thoroughly. There have been different responses to this objection, which aims to show that one can still act rationally even if the reasoning is not based on knowledge, but we aim to set as much of this debate aside as possible. Our point, rather, is that if there is something like a knowledge norm, then it is clear what is problematic about these cases: there was a violation of the norm, and that led to a faulty reasoning process.

Furthermore, a knowledge norm is not required for our analysis to work, though it works well if a knowledge norm holds. Weiner (2005) proposes a truth norm and (Henning, 2021) an epistemic modal norm closely related to it, and on both norms there is a violation in the cases where a user acts on erroneous information provided by the system. Brown (2008) rejects the knowledge norm but maintains a context-sensitive norm that is sometimes weaker than the knowledge norm and sometimes stronger, and can function in the same way in this analysis (at least for cases where the stakes are sufficiently high).

The main issue for our analysis is the few philosophers who suggest a norm based on justified belief (Littlejohn, 2009) or warrant (Gerken, 2011). We assume that in the situations where users take the outputs of the system into their practical reasoning they are justified to believe these outputs (e.g. have been correctly told by the developers that the system is overall reliable, and so would not violate these norms if the system output is wrong). Of course, under exactly which conditions users are justified to trust a (ML) system is an open research question (cf. Durán & Jongsma, 2021; Ferrario, 2021; Ferrario & Loi, 2021; Jacovi et al., 2021) and so this assumption may be wrong. Still, we mention the assumption because only in that case does our account face a difficulty, as on the justified belief norm it is unproblematic to follow the outputs of a trustworthy system even if they are wrong. It wouldn't be a problem if the user isn't justified to believe the output (i.e. if the system isn't trustworthy), as then this norm of practical reasoning is also violated and users shouldn't use the outputs in practical reasoning whether the output is correct or not. The question then is how our analysis holds up in the situation where users *are* justified to believe the outputs, whatever the conditions may have to be met to reach this state.

We think that even then we can say that the ultimate goal of these sociotechnical systems is to foster good decisions. So, if there is conflicting information, in the form of defeaters (our proposed solution to the issue of incorrect system output), then that will remove or reduce the warrant or justification of the belief. A user would violate the norm if conflicting information is willfully ignored, but does not do so if the user was not aware of it at the time of the decision. This seems fair, and tracks with the intuition that hiring managers would have attracted less blame if they follow the AI advice when it is wrong, as time pressure makes checking conflicting information difficult. It can also maintain the idea that the operators of the Vincennes did something wrong: they had conflicting information, which would have removed their warrant/justification, readily at hand but failed to pick it up. There may not be a norm violation here, but there is a clear case where users should have done something more to ensure that they had the right basis for practical reasoning and decision making. No matter what the norm for practical decision making is, then, the case of incorrect system output is readily analysed by appealing to it. For better practical decision making, we should aim for the situation where a user has relevant conflicting information in cases where the system output is erroneous.

Yet, how does one design for that? We think this challenge can be approached by looking at defeaters, i.e. pieces of information that reduce one's justification to believe a claim, and their two main types.

3 Defining Defeaters

In both of the examples in the introduction there is information that, had it been available at the time of the decision, would have changed the outcome. This information is readily classified as consisting of defeaters: propositions such that they lower our credence in the claim of the system. For hiring managers this could be a warning that the algorithm is less reliable for a minority to which the candidate belongs, or has been referred by an internal employee. In the case of the USS Vincennes the commercial flight log is an obvious defeater, as is the information that the plane was climbing rather than descending. Just as relevant would have been a warning that the operator forgot to reset the range of the identification system, which would defeat the reliability of the returned ‘military aircraft’ tag. Still, designing for defeaters is no easy task, so we start by discussing the two main types of defeaters as a way to structure the more applied Sect. 4.

Our point of departure is the distinction made by Pollock and Cruz (1986) between *rebutting* and *undercutting* defeaters. Informally: we start in a situation where proposition P is supported by evidence E. A rebutting defeater R directly speaks against the truth of P. In contrast, an undercutting defeater U reduces E’s evidential support for P. That means that the commercial flight log was a rebutting defeater for the claim that the approaching aircraft was a military plane, whereas a message that the identification system was not reset would have been an undercutting defeater (as it raises doubts about the reliability of the identification system). Similarly, a warning message about biases in the algorithm would be an undercutting defeater for a hiring manager, whereas a referral or exceptionally good reference could act as rebutting defeaters.

To make this more precise the formal definitions of Kotzen (2019) might help (his ‘opposing’ is our more standard ‘rebutting’):

> D is an opposing defeater for the evidence that E provides for H just in case [p(H| E & D) < p(H|E)] & [p(H|D) < p(H)]. (Kotzen, 2019, p. 221)
> D is an undercutting defeater for the evidence that E provides for H (relative to background information K) just in case *dc*(E, H, K) > *dc*(E, H, K & D) (Kotzen, 2019, p. 224)

where *dc* (E, H, K) is the degree to which evidence E confirms hypothesis H given background knowledge K (where the defeater acts as additional background knowledge). Kotzen (2019, p. 225) offers several formal definitions of this degree of confirmation, but prefers the log likelihood ratio:

$$dc = log\left(\frac{p(E|H\&K)}{p(E|\neg H\&K)}\right)$$

Alternatively, the work of Mayo (1996, 2018) can be used to define defeaters in a frequentist framework, using her notion of severe tests. As the basis for her framework is the following:

> Severity Requirement: for data to warrant a hypothesis H requires not just that
> (S-1) H agrees with the data (H passes the test), but also
> (S-2) with high probability, H would not have passed the test so well, were H false. (Mayo, 2018, p. 92)

The hypothesis in the current situation is again the output of the system. This has some support, as the system is (we will assume) reliable generally speaking. So, H has in our scenario already passed one test by being output by a reliable system.

Defeaters can be viewed as applying either to S-1, by presenting additional data which does not agree with H (rebutting defeaters) or to S-2, by presenting evidence that H could have passed the general test—being the output of the system—even if H were false (undercutting defeaters). While undercutting defeaters are hard to construe as new severe tests (as they do not confirm or disconfirm H), rebutting defeaters can be seen as new severe tests. They bring in additional evidence, and the work of Mayo (2018) might help with the inevitable trade-offs in situations where different severe tests present different outcomes.

We should note here that these are not the only two types of defeaters suggested in the literature. Kotzen (2019) already discusses hybrid variants, but Dutant and Littlejohn (2021) offer a list of eight suggestions, with rebutting and undercutting defeaters as the first two items. Muñoz (2019) adds the notion of a disqualifier, which makes more indirect evidence obsolete (and relying on the more indirect evidence irrational). Furthermore, there is an ongoing philosophical discussion on whether defeaters have some normative status (e.g. Goldberg, 2017; Lackey, 2006) or are psychological states (Bergmann, 2006). While these discussions, and especially those on further types of defeaters, may be relevant for further work on designing for defeaters, we leave them aside here. As we aim to show, the basic distinction between rebutting and undercutting captures a lot of design options already, and is helpful enough to capture the existing methods for detecting when an algorithm produces the wrong output. We therefore choose for the simplicity of the sparser classification of defeaters.

Given this basic classification we next turn to our more practical contribution. The leading question for the paper is: how do we help users avoid following the mistakes of an automated system? Our main suggestion is that we do so by actively designing the sociotechnical system such that it looks for defeaters and, if any are present, shows these to the user. One way to structure this idea is by conducting a Failure Mode and Effects Analysis (FMEA), a method already commonly used for anticipating mechanical failures.

4 Designing for Defeaters

Conducting an analysis of the ways in which a (sociotechnical) system can go wrong is already standard practice in engineering, under the name Failure Mode and Effects Analysis (FMEA, see e.g. Stamatis, 2003). It has made a wide variety of systems incredibly reliable, and so offers a promising basis for design aimed at reducing the consequences of errors in automated systems related to the information provided to users. It has, moreover, also been suggested as a basis for AI ethics audits (Raji et al., 2020). The basic idea is that one identifies the different ways in which a product, process, service or machine might fail, the likelihood and consequences of those failures and preventative actions to take. We think the same can readily be applied for the information provided in sociotechnical systems.

To see how this might work, consider the case of the USS Vincennes again. As part of the design process one would then consider: what if the tagging system incorrectly identifies an aircraft as 'military' (or vice versa)? Potential consequences are very serious, so defeaters should be designed into the system. This was done, with the commercial flight log being available, but as the room was dark and the ship made sharp turns during the decision process it was difficult to get the right information from this log. Similarly, the information whether the plane was ascending or descending was available, but on a screen on the other side of the room. So this is a case where the initial design step (ensuring defeaters are available) is certainly met, though it turned out that they were in practice too difficult to obtain to prevent a mistaken decision. As we will see below, it is however far from standard that defeaters have been included in the design of sociotechnical systems. So what should one pay attention to in the case of automated systems? We think a high-level classification of the different ways in which a sociotechnical system may lead to erroneous output is a good starting point. Most of these are different cases of overfitting, as that is the source of most mistakes made by ML systems. As such, defeaters are certainly not the only way to tackle this problem, as e.g. correctly identifying the underlying causal relations (in the spirit of Pearl (2000)) might in the long run prevent systems from making these mistakes. While we encourage such efforts, we also consider it unlikely that all such mistakes can be prevented in the short run. Hence the cataloguing of ways in which systems can currently go wrong, and the choice to present different causes of overfitting separately, as they will also be spotted in different ways. Consequently, they give rise to different defeaters.

First, there is the scenario where a system returns an incorrect output simply because it is not completely accurate. A classification algorithm may, for example, only classify 90% of cases correctly on the validation set, so even if the input is relevantly similar there is a chance that the output is incorrect. What happens in that case, and what information might help? Both undercutting and rebutting defeaters can be relevant. Consider the simple example of a convolutional neural network analysing video clips for the presence of cats or dogs, based on the visual information. An undercutting defeater might be a saliency map (e.g. Fong & Vedaldi, 2017) showing that for the output 'cat' the algorithm weighed pixels showing grass most heavily. This undermines the confidence one should put in the correctness of the algorithm, even though it might still happen to be right in this instance. Similarly, ongoing work in uncertainty quantification (Abdar et al., 2021 for a review) aims to offer information on how likely a certain algorithmic output is correct and can provide undercutting defeaters in the cases where this certainty is low. A rebutting defeater, on the other hand, might be a separate system that detects barking on the same video clip, indicating that a dog is present (and thus conflicting with the output of the visual classifier). Both approaches will give the user a reason to doubt the outcome of the algorithm, possibly presenting failures in case the output is wrong.

Second, there is the possibility of covariate shift and concept drift, where the data to which the algorithm is applied differs (sufficiently) from the training data and makes the model inaccurate as a result. This difference can either result from a failure to sample the dataset representatively, or it can result from the world changing, making the training data unrepresentative of the new situation (even though it was representative for the old situation). This is a worrying case as without instructions lay users seem to rely *more* on algorithms in these cases than otherwise (Chiang & Yin, 2021). We distinguish this case from the first, as different types of defeaters apply, and systems can have a degree of inaccuracy on the training dataset too without having problems with covariate shift. So, aside from the defeaters mentions for general inaccuracy, there is a second class here associated with this other reason for incorrect model output. These defeaters will aim to signal that covariate shift has occurred, and will generally be undercutting defeaters. For example, one might have a signal that the input data is an outlier compared to the training data (which helps users make better decisions in these cases; Poursabzi-Sangdeh et al., 2021). In the case of rebutting defeaters, these might apply because covariate shift can also mean that the weighing of different features is no longer correct for the data to which it is currently applied. For example, if a system gives employability estimates then the situation might shift due to the introduction of programs designed to help job seekers with specific skill sets. If the model is not updated to reflect this, then the fact that a job seeker has the relevant skills becomes a rebutting defeater for the model output: their qualification for the program is a reason to think they will be more employable. In the case of covariate shift, then, we might have undercutting defeaters indicating that the data distribution has changed or that a specific instance is an outlier, or rebutting defeaters based on the source of the changes that make the model less accurate.

Third, the output of a system may be wrong because it is missing relevant features. Again, we distinguish this from general inaccuracy because there is a more specific reason for the error here rather than just the fact that algorithms are generally not 100% accurate. Feature selection is an important part of system design, but one cannot guarantee that all the important features have been found. Additionally, during deployment the situation may change resulting in new features becoming relevant that were not important yet when the system was designed. For example, a system designed to estimate the employability of citizens asking for benefits might only ask for occupation, education, employment history and county (as some systems in the US did; Desiere et al., 2019). This can miss relevant features, such as additional skills acquired. For example, a job seeker who has spent time acquiring coding skills may well be more employable than the model estimates, if ‘education’ only contains school degrees. Conversely, if the job applicant also has a heavy burden of care, reducing the number of hours he/she can work, the employability score may be too high, giving too rosy a picture of how easy it will be to find a suitable job. Though it will likely be difficult for public officials to spot such missing information on their own, let alone for them to adjust the model outcome based on this, we do suggest that those affected by the decision should be offered a way to present such additional information (as rebutting defeaters), as part of the wider attempts to make decisions contestable.

Fourth, one might consider (the socio-political notion of) bias as a way in which an automated system may give incorrect outputs. Consider a case where an algorithm suggests to reject a female applicant (which can be the case due to statistical biases, because e.g. fewer female applicants were included in the training data, or because gender correlated with the hiring decision in the data) even though it has suggested that a male applicant with the same qualifications be hired. This earlier recommendation can then act as a rebutting defeater for the case of the female applicant. The absence of a clear reason for her rejection, other than her gender, is itself a reason to doubt the accuracy of the decision. Now, this is a somewhat different case than the previous three. The underlying reason why bias is problematic is normative: we think that decisions *should not* be based on such factors. That does not automatically mean that outputs that are biased are also more likely to be incorrect, which is the epistemic connection we need for information to count as a defeater. Still, we consider bias as an appropriate type of system failure as it is plausible that biased systems are generally less accurate. The very point is that these protected attributes are often irrelevant: the information systems aim to provide (e.g. employability, risk of fraud) typically does not causally depend on gender, sexual orientation or nationality in the world even if the system output may depend on these features. Statistical correlations will appear in the data, especially since we are not free from biases, but they do not reflect causal relations. That mismatch, which links naturally to the project of Pearl (2000) mentioned above, to work out causal models to such an extent that algorithms can use proper causal reasoning. As the cases of socio-political bias are instances of faulty causal reasoning they too signal that something is wrong with the outcome of the algorithm. Signals of such bias give therefore, to our mind, sufficient reason to think that there is also something epistemically wrong with a decision that is based on a biased process.

We thus count bias as a source for defeaters, and not just as a different type of reason to question algorithmic output. Specifically, counterfactuals that show that the outcome had been different if e.g. race or gender was changed may help to produce rebutting defeaters. This can be implemented in different forms, such as flipsets (Spangher et al., 2018; a list of changes that can flip the prediction of a classifier) or with other tools from the algorithmic recourse literature (Lyons et al., 2021). Furthermore, Dodge et al. (2019) have found that offering such counterfactuals on protected features can help expose bias to users of a system. They are thus a good candidate for offering rebutting defeaters, as there are not only tools available but these tools also seem to help users detect problematic cases.

Fifth, and by far the hardest to design for, systems may lead their users astray because the statistical correlations they are based on are spurious. This is closely related to the case of bias, but is a wider issue of the possibility to find statistical correlations in datasets that do not reflect causal dependencies in the world. We can expect such systems to lead users astray, as in the case of that developed by Wu and Zhang (2016), to predict criminality based on images of people's faces. There is no scientific evidence that one can infer criminality on such a basis (phrenology has been debunked long ago), and the fact that any such link between facial features and criminality is missing is an undercutting defeater for systems attempting to do just that. The same issue occurs for systems that try to infer emotions based on people's facial expressions (Barrett et al., 2019). However, this heavily depends on outside evidence that the system is based on a spurious correlation in the dataset, so we see no tools easily available to detect this, other than to consult the scientific literature before designing such a system.

In these ways, we suggest that designers of sociotechnical systems look at the different reasons why system output might be wrong (or more accurately, might lead to bad decisions by the user) and consider what information might be helpful to the user to avoid those bad decisions. To recap, these are:

1. System inaccuracy
2. Covariate shift
3. Missing features
4. System bias
5. Spurious correlations

We may not have managed to be exhaustive in our list of suggestions here, but hope that the idea is clear enough to pick up in the design process. Finally, we want to add one further consideration with respect to defeaters. So far we have presented them as information to give in case the algorithm output is wrong. I.e. notify the user when a defeater is present. On the other hand, it might be equally helpful to notify a user that nothing is amiss, i.e. that no defeaters have been found. Klein (2014) introduces this idea as there being 'confirmers' (that the system is reliable in this case and that other sources of information lead to the same conclusion). As we focus here on the cases where it goes wrong, we will not discuss this idea much further, but it could be that the absence of defeat is likewise helpful information for users of sociotechnical systems. That being said, we lastly turn to a case study to further illustrate how our idea of designing for defeaters might work in practice, and in particular how it might apply to ML systems.

5 Case Study: The Dutch Childcare Benefits Scandal

In 2011 the Dutch tax agency decided to introduce a sociotechnical system to carry out better checks before benefits were paid out. The reason for introducing the system is to reduce the possibility of fraudulent applications, such as the later discovered 'Bulgarian fraud' which consisted of Bulgarian criminals unjustifiably collecting rent and healthcare benefits. The system was extended in 2013 with an algorithm that assigns benefits applications a risk score, and in particular labels some of them as potentially fraudulent. To do so, it uses the trove of data available to the tax authority. However, it gradually became clear that this algorithm (frequently followed by the employees making the final decisions) was mistakenly classifying many people with dual nationality as (potential) fraudsters. This realization led to the tax agency, in June 2019, deciding to no longer include dual nationality in the database queries (van Huffelen, 2020). Moreover, in July 2020, the Dutch Data Protection Authority "Autoriteit Persoonsgegevens" concluded that the processing of dual nationality by the tax authorities in risk selection models was unlawful and discriminatory. This affair played an important role in the Dutch cabinet resigning in 2021.[2]

Further investigation revealed that while designers had chosen to make an official responsible for the final decision on whether or not to designate an applicant as suspicious, the official was heavily epistemically dependent on the system. The

[2] Another important factor in the resignation of the cabinet was the unnecessarily harsh approach to fraud by the "Combiteam Aanpak Facilitators", aimed at, for example, guest parent agencies.

management team had purposely decided to give the official as little information as possible, with the system's only output being that the application was suspicious or not. No supporting evidence for this claim was presented and operators had no alternative sources of information regarding the correctness of the system classification.[3] Moreover, it was discovered that even after the Dutch Data Protection Authority concluded that the use of dual nationalities was immoral and illegal, this data kept being requested from databases and used, probably perpetuating the bias against dual nationals (van Huffelen, 2020).

How could this have been improved upon? The official here is supposed to decide whether an applicant is to be investigated. In the current setting, clearly, the official is epistemically dependent on the system due to the lack of options for independent verification, and so will be steered wrong if the system output is incorrect (as it was for many dual nationals). We therefore suggest that at the design stage defeaters should have been considered, as these could have helped alleviate the problematic situation that resulted.

One general challenge in doing so will be the question of trade-offs in presenting defeaters. Users of the system will have to integrate different sources of evidence when receiving defeaters or may end up in a situation with no clear right answer (e.g. if there are only undercutting defeaters, showing the system to be unreliable without giving positive evidence for any decision). It could turn out that users of the system have a hard time doing so, and end up making worse decisions in a range of cases. Furthermore, having defeaters implies having to spend more time on a decision. So what is the threshold one puts up before presenting a defeater? This probably varies with the stakes, as we want to have a higher certainty before ordering missile strikes than before rejecting a job candidate. Still, some decision has to be made on when a defeater is relevant and strong enough. We'll set aside this question here to focus on what defeaters might be available, and merely note that the statistical frameworks from Sect. 3, such as that of Mayo (2018) might be helpful to make these trade-offs. That leaves us room to focus on the types of defeaters relevant to the Dutch tax agency case. We'll go through these based on the classification provided in Sect. 4. To start with, there are mistakes based on general system inaccuracy.

5.1 Defeaters Based on System Inaccuracy

ML algorithms do not always base their decisions on features that are actually important, as with the example of the decision 'this picture contains a cat' being

[3] To be precise, the entire assessment consists of two stages. The first stage assesses whether an applicant is a potential fraudster. During this stage, no information is given to the operator as to why the system sees an applicant as a potential fraudster. However, once the operator has decided that the applicant should be investigated, she has the opportunity to check all the information herself in order to find out the details. However, this is a time-consuming process in which all the data is in principle relevant. In practice, applicants who are labelled "possible fraudster" in the first phase are often viewed very critically and small errors in the application can lead to penalties. In this paper we focus on the first stage of the assessment.

based on pixels showing grass. Similarly, the tax agency algorithm could have been based (and in fact was, at least in the case of nationality) on features that are in fact irrelevant. Though we will discuss the dual nationality case under the heading of Bias, we do think that feature importance methods can be relevant generally speaking. Users could be presented with these on every application, but to prevent information overload and focus the attention of officials on preventing bad outcomes we suggest focusing on applications that are classified as potentially fraudulent. In such cases the most important features for the algorithm's decision may be listed, so that it is possible to check whether these make sense. A challenge, though, could be the non-linearity of ML algorithms that makes it hard to evaluate whether the importance of 'age' is problematic or not.

More information could be of help here. If, in addition, uncertainty quantification methods are used (e.g. Pearce et al., 2018), then an official would justifiably doubt the system more in cases where the uncertainty is high. If, moreover, this is coupled with unintuitive feature importance scores, then the official will have salient undercutting defeaters and be in a good position to doubt the system. Again, we suggest focussing on the applications labelled as suspicious to prevent officials from having to sift through too many indicators. Furthermore, it can be worthwhile to maintain a log of the important features that underlie (potentially) wrong decisions, so that an overview emerges of the situations in which the algorithm performs poorly. This overview can then be used for further undercutting defeaters, by indicating that a combination of features is found in the application that has led the algorithm astray in earlier situations.

Moreover, it was possible to design for rebutting defeaters early on based on an analysis of the cases of fraudulent activities that were part of the training data. It was known that the Bulgarian criminals that were the reason for implementing the system encouraged Bulgarians to register at a fake Dutch address and wrongfully apply for benefits. So, if data from external systems indicates that the applicant has been registered at the address that is mentioned in the application and has paid taxes for several years, then this is evidence that an application is innocent. It means that the application is likely not from one of these fraudsters, and moreover presents a positive reason to think that the applicant is honest. One may worry, though, that it leads to a large number of false negatives; fraudulent applications that are approved. Users might simply approve any application they see where the address checks out and the taxes have been paid properly. A crucial question here seems to be the extent to which this information has been incorporated by the automated system. If it has been weighed up, but there are sufficiently many other indicators to suspect fraud, then it could indeed lead to false negatives. Yet if, as with our barking example, it is system-independent *and* a strong indicator for a particular outcome, then it should be presented. Possibly it is better to send such applications on for further (human) review, rather than letting an official accept them directly, but we do think that it is important to consider such information as well.

A different question, though related, is whether this external information that is being provided is still a good indicator of, in this example, an honest application for benefits. Perhaps the Bulgarians have shifted methods and are no longer using fake addresses. This brings us to the case of covariate shift, as both our suggested rebutting defeater and the system will be affected by it.

5.2 Defeaters Based on Covariate Shift

Our suggested tool for undercutting defeaters based on covariate shift from Sect. 4, outlier detection, will likely apply. Though we have been unable to verify this, it seems likely that there were some outliers in the data distribution to which the risk classification algorithm was applied. We can expect that in these cases, as typically holds, the algorithm is less reliable and so an undercutting defeater would be provided. So, outlier detection is an option that can be investigated, but focussing on specific cases of covariate shift such as the changing methods of the Bulgarian criminals offers—in our opinion—a more promising way to mitigate the effects of epistemic dependence on the sociotechnical system.

For, after the fraud by Bulgarian criminals came to light they shifted their work to other regions and methods. This is a change in (criminal) behavior that has the effect that the data to which the algorithm is applied differs from the training data and makes the model inaccurate as a result. This covariate shift was known to many people working with the system, but the model was not updated to reflect this. Feature importance signaling can be used to make the operator aware of this kind of mistake in the model, especially if this implicit knowledge of the changing methods of Bulgarian criminals is made explicit by a message accompanying high feature importance of an applicants' Bulgarian nationality. We can then present the user of the system with an undercutting defeater, by signalling that the system may have classified this application based on statistical correlations that no longer hold.

Focusing on such specific changes in the real world that invalidate the model predictions can also work as a method for finding rebutting defeaters. One of the features used as indicators of fraud was the distance between the applicant's address and the address of the childcare provider. After a few years it became well-known that this feature was a reason for the tax agency to look more closely at the application. We have not been able to verify this, but it is likely that people with bad intentions chose a childcare provider that is closest to their own address in the application, meaning that a small distance between addresses was an indicator for fraud. If, instead, fraudsters systematically started to pick childcare providers that were further away, then having an address near to the provider is a rebutting defeater for the system output. Not only does it give us a reason to doubt that the system is accurate (as it is based on data from before the change in locations picked by fraudsters), the knowledge that fraudsters avoid nearby childcare providers turns it into a positive reason to think that the applicant is honest. Naturally, the strength of such a defeater will depend on how strong the correlation between a small distance and the absence of fraud is. So, such cases need to be examined with care, but can—if the correlation is strong enough—at least provide reason for a more thorough review. It is, after all, this kind of missing information that can lead algorithms further and further astray. That is one reason why it is, additionally, relevant to have a mechanism for the identification of missing features.

5.3 Defeaters Based on Missing Features

The system that was actually implemented did not allow for applicants to supply additional information to challenge the decision labelling them as potential fraudsters in the first phase of the review process. However, if the system had been designed for such contestation of the decisions then this would also have lead to the provision of rebutting defeaters. Applicants might have supplied additional information, quite possibly not considered by the system, to prove that their application was honest. For operational reasons, not all indicators have been made public. We, therefore, have not been able to find exactly what information the system did and what it did not take into account. It seems plausible, though, that there will have been information that the unfairly suspected applicants possesed which shows that they did not commit fraud. Whether this was a positive confirmation from the childcare provider or proof that they live at the stated address, such documents could have been looked for. In particular, the system might have indicated to applicants looking to contest the decision what features weighed most heavily in the decision, and so what type of information they should have provided to convince the tax agency that it had made a mistake. Although this process will require more human review, we think that this is ultimately a good thing in the high stakes context for which the risk classification algorithm was developed. It might have mitigated the bias that was the ultimate reason for abandoning the system, though the steps above alone are not sufficient to fully signal such crucial mistakes. Therefore, we finally turn to the defeaters to be designed based on (possible) system bias. We omit the spurious correlation case as, like we mentioned in Sect. 4, it does not seem to apply here.

5.4 Defeaters Based on System Bias

Bias in the system played an important role in the resignation of the Dutch cabinet. Morally, this is also one of the more worrying ways in which an algorithm might present us with erroneous/undesirable output. So, we think any attempt to counter this will help, and in particular (as discussed) view defeaters as a promising way to control and hopefully prevent this type of error. As discussed, flipsets (or other counterfactual methods) can be used to inform the user about biases in a system. Spangher et al. (2018) use these flipsets to identify "actionable" features that the user should change in order to alter the system's decision to a more favourable one. By altering this focus from actionable to "protected" features, identifying whether there are any such features that would lead to a change in outcome, biases can be signalled. Typical protected features are those characteristics that are protected in the constitution under anti-discrimination law. Chapter 1, article 1 of the Dutch constitution prohibits discrimination on the basis of religion, philosophy of life ("levensovertuiging"), political affiliation, race and gender. The Dutch Data Protection Authority also deemed the use of the applicant's nationality morally and legally problematic for the system under discussion. This feature should therefore also be classified as "protected". If the system finds that a flipset can be generated using only protected features, or with sufficiently small changes to non-protected features along with larger changes to protected features, then this should be signaled to the operator as a rebutting defeater. In this particular context, we expect that dual nationality would have surfaced as such a feature, as it has by now emerged that exactly that feature led to a large number of false positives.

As such, we have a broad selection of defeaters that might have helped. From feature importance (generally and specifically focussed on protected features) to outlier detection, from the information that Bulgarian criminals changed their methods to potential proofs of non-fraudulent behaviour via e.g. documented proof from the childcare provider, a wide range of information could have been used to spot situations where the system would otherwise steer officials wrong. We are convinced this information can help, and presented some empirical studies showing exactly this in Sect. 4, but at the same time realize that the implementation will be a challenge. Most of all, then, we hope that this focus on information that might show a system is incorrect will be taken on board in the design process, leading to fewer issues stemming from our epistemic dependence on these systems.

6 Conclusion

Users of sociotechnical systems are often in the difficult situation of having to make a decision based on system output while, at the same time, being unable to independently verify the correctness of that output. They're epistemically dependent on the system and therefore may make bad decisions, and likely violate norms of practical reasoning, in cases where the system makes an error. As we have argued, the main difficulty here is that there is no information provided to these users to indicate to them when the system is less reliable (in which case it is an undercutting defeater) or when there is conflicting information available (a rebutting defeater). It is this information, these defeaters, that can help reduce mistakes that naturally result from the epistemic dependence on an imperfect system. They help limit the epistemic dependence on the main algorithm (e.g. the risk classification algorithm of the Dutch tax agency) and though the user is still epistemically dependent on the sociotechnical system as a whole, the defeaters present the user with more nuanced outputs.

To shift this from pure theory to a more practical design question we have therefore looked at what types of defeaters a designer might implement. We used a method similar to Failure Mode and Effects Analysis, which is used to mitigate the aviation errors we opened the paper with, to identify in what ways an algorithm might yield incorrect output. Each of these five leads to distinct types of defeaters, most of which apply to the case study we presented in Sect. 5. Now, as we mentioned in several places, defeaters will still be tricky to design. There are questions on how strongly to weigh each defeater, determined by the degree to which they lower the probability that the algorithm is reliable/the output is true, and the methods proposed (such as counterfactual methods) still come with technical challenges. We think, however, that the general goal of designing sociotechnical systems for defeaters is worthwhile and important to keep in mind.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P. W., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., & Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*. https://doi.org/10.1016/j.inffus.2021.05.008

Adam Carter, J. (2017). Intellectual autonomy, epistemic dependence and cognitive enhancement. *Synthese, 197*, 2937–2961.

Adam Carter, J. (2021). Epistemic autonomy and externalism. In K. Lougheed & J. Matheson (Eds.), *Epistemic autonomy*. Routledge.

Barrett, L., Adoplhs, R., Marsella, S., Martinez, A., & Pollak, S. (2019). Emotional expressions reconsidered: Challenges to inferring emotion from human facial movements. *Psychological Science in the Public Interest, 20*(1), 1–68.

Bergmann, M. (2006). *Justification without awareness*. Oxford University Press.

Brown, J. (2008). Subject-sensitive invariantism and the knowledge norm for practical reasoning. *Nous, 42*(2), 167–189.

Chan, S. (2017). Stanislav Petrov, Soviet Officer who helped avert nuclear war, is dead at 77. *The New York Times*. Retrieved September 18, 2017, from https://www.nytimes.com/2017/09/18/world/europe/stanislav-petrov-nuclear-war-dead.html

Chiang, C., & Yin, M. (2021). You'd better stop! Understanding human reliance on machine learning models under covariate shift. In *The 13th ACM web science conference*, June 2021.

Desiere, S., Langenbucher, K., & Struyven, L. (2019). Statistical profiling in public employment services: An international comparison. OECD Social, Employment and Migration Working Papers, 224. OECD.

Dodge, J., Liao, Q., Zhang, Y., Bellamy, R., & Dugan, C. (2019). Explaining models: an empirical study of how explanations impact fairness judgment. In *Proceedings of the 24th international conference on intelligent user interfaces* (pp. 275–285).

Durán, J., & Jongsma, K. (2021). Who is afraid of black box algorithms? On the epistemological and ethical basis of trust in medical AI. *Journal of Medical Ethics, 47*(5), 329–335.

Dutant, J., & Littlejohn, C. (2021). Defeaters as Indicators of Ignorance. In M. Simion & J. Brown (Eds.), *Reasons, justification, and defeat* (pp. 223–246). Oxford University Press.

Erlei, A., Nekdem, F., Meub, L., Anand, A., & Gadiraju, U. (2020). Impact of algorithmic decision making on human behavior: Evidence from ultimatum bargaining. In *Proceedings of the AAAI conference on human computation and crowdsourcing* (Vol. 8(1), pp. 43–52).

Ferrario, A., & Loi, M. (2021). The meaning of "Explainability fosters trust in AI". SSRN 3916396.

Ferrario, A. (2021). Design publicity of black box algorithms: A support to the epistemic and ethical justifications of medical AI systems. *Journal of Medical Ethics*. https://doi.org/10.1136/medethics-2021-107482

Fong, R., & Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *2017 IEEE international conference on computer vision (ICCV)* (pp. 3449–3457), Venice, Italy, 2017.

Gerken, M. (2011). Warrant and action. *Synthese, 178*, 529–547.

Goldberg, S. (2017). Should have known. *Synthese, 194*, 2863–2894.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local rule-based explanations of black box decision systems. arXiv preprint. arXiv:1805.10820.

Hardwig, J. (1985). Epistemic dependence. *The Journal of Philosophy, 82*(1), 335–349.

Hawthorne, J., & Stanley, J. (2008). Knowledge and action. *Journal of Philosophy, 105*(10), 571–590.

Henning, T. (2021). An epistemic modal norm of practical reasoning. *Synthese, 199*(3–4), 6665–6686.

Humphreys, P. (2009). The philosophical novelty of computer simulation methods. *Synthese, 169*(3), 615–626.

Jackson, A. (2012). Two ways to put knowledge first. *Australasian Journal of Philosophy, 90*(2), 353–369.

Jacovi, A., Marasović, A., Miller, T., & Goldberg, Y. (2021). Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in AI. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency* (pp. 624–635).

Klein, R. (2014). Where there are internal defeaters, there are "confirmers." *Synthese, 191*, 2715–2728.

Kotzen M. (2019) A Formal Account of Epistemic Defeat. In: B. Fitelson, R. Borges & C. Braden (Eds.) *Themes from Klein*. Synthese library (Studies in epistemology, logic, methodology, and philosophy of science) (Vol. 404, pp. 213–234). Springer.

Lackey, J. (2006). Learning from words. *Philosophy and Phenomenological Research, 73*, 77–101.

Lehrer, K. (1990). *Theory of knowledge*. Routledge.

Littlejohn, C. (2009). Must we act only on what we know? *Journal of Philosophy, 106*(8), 463–473.

Logg, J., Minson, J., & Moore, D. (2019). Algorithm appreciation: People prefer algorithmic to human judgment. *Organizational Behavior and Human Decision Processes, 151*, 90–103.

Lyons, H., Velloso, E., & Miller, T. (2021). Conceptualising contestability: Perspectives on contesting algorithmic decisions. *Proceedings of the ACM on Human-Computer Interaction, 5*(CSCW1), 1–25.

Mayo, D. (1996). *Error and the growth of experimental knowledge*. The University of Chicago Press.

Mayo, D. (2018). *Statistical inference as severe testing: How to get beyond the statistics wars*. Cambridge University Press.

Mehta, N. (2016). Knowledge and other norms for assertion, action, and belief: A teleological account. *Philosophy and Phenomenological Research, 93*(3), 681–705.

Mueller, A. (2021). The knowledge norm of apt practical reasoning. *Synthese, 199*(1–2), 5395–5414.

Muñoz, D. (2019). Defeaters and Disqualifiers. *Mind, 128*(511), 887–906.

Pearce, T., Brintrup, A., Zaki, M., & Neely, A. (2018). High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International conference on machine learning* (pp. 4075–4084).

Pearl, J. (2000). *Causality: Models, reasoning and inference*. Cambridge University Press.

Pollock, J., & Cruz, J. (1986). *Contemporary theories of knowledge*. Rowman and Littlefield.

Poursabzi-Sangdeh, F., Goldstein, D., Hofman, J., Wortman Vaughan, J., & Wallach, H. (2021). Manipulating and measuring model interpretability. In *Proceedings of the 2021 CHI conference on human factors in computing systems* (pp. 1–52).

Raji, I., Smart, A., White, R., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D., & Barnes, P. (2020). Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *FAT* '20: Proceedings of the 2020 conference on fairness, accountability, and transparency* (pp. 33–44), January 2020.

Rochlin, G. (1991). Iran Air Flight 655 and the USS Vincennes. NATO ASI Series (Series D: Behavioural and Social SciencesIn T. R. La Porte (Ed.), *Social responses to large technical systems.* (Vol. 58). Springer.

Rooksby, E. (2009). How to be a responsible slave: Managing the use of expert information systems. *Ethics and Information Technology, 11*, 81–90.

Spangher, A., Ustun, B., & Liu, Y. (2018). Actionable recourse in linear classification. In *Proceedings of the 5th workshop on fairness, accountability and transparency in machine learning*.

Stamatis, D. (2003). *Failure mode and effect analysis: FMEA from theory to execution*. American Society for Quality, Quality Press.

van den Hoven, J. (1998). Moral responsibility, public office and information technology. In I. Snellen & W. van de Donk (Eds.), *Public administration in an information age: A handbook* (pp. 97–112). IOS Press.

van Huffelen, A. C. (2020). *Kamerstuk II 2019/20, 31 066*, Nr. 683. https://zoek.officielebekendmakingen.nl/kst-31066-683.html

Weiner, M. (2005). Must we know what we say? *Philosophical Review, 114*(2), 227–251.

Williamson, T. (2005). Contextualism, subject-sensitive invariantism and knowledge of knowledge. *The Philosophical Quarterly, 55*(219), 213–235.

Wu, X., & Zhang, X. (2016). Automated inference on criminality using face images, pp. 4038–4052. arXiv preprint. arXiv:1611.04135