Opacity, Big Data, Artificial Intelligence, and Machine Learning in Democratic Processes (DRAFT)

Ramón Alvarado
University of Oregon

The first decades of the 21st century saw an initial hubris regarding the prowess of computational methods such as big data, artificial intelligence, and more recently machine learning. However, the opacity related to these computational methods—the inability to access, examine or even identify relevant aspects of their inner workings—is consequently at the front and center of recent concerns. Understanding the different kinds of opacity and the ways in which they are present in these computational methods is a fundamental step towards understanding the challenges they present to democratic processes in general.

The general argument of this chapter is the following: epistemic opacity, and particularly its distinct forms, must be taken into account in order to accurately assess the profound challenges that computational analytic methods such as big data, artificial intelligence and machine learning, pose to democratic processes. This is because it is important to understand that a significant aspect of the novel challenges brought about by these computational methods is of an epistemic nature. That is, knowing participants of democratic processes are being left behind as technologies used in the various facets of democratic processes continue to grow in complexity and size beyond what is humanely tractable. If we take a free and knowing agent as a fundamental player in the dynamics of democratic processes, the opacity of computational methods such as big data, machine learning and artificial intelligence are in fact a significant challenge to democratic processes as a whole.

Democratic processes

A democratic process can be many things. It can be a parliamentary or congressional legislative procedure within a democratic country. It can also be the act of voting for members of such institutions by individual citizens. In between these two types of democratic processes there are several other interactions that also count towards or against the aims of democratic processes. For example, the dissemination of information through electoral campaigns that are now often mediated through diverse computational methods and general digital infrastructures that—as I will show below—due to their complexity and scale push an informed citizenry further and further away from a thorough understanding of the core mechanisms by which these processes function. As we will see, the opacity inherent to computational methods is not always a direct result of malicious intent. In fact, in some cases opacity is generated simply by the inherent size and/or complexity of the technologies underlying such processes. However, the effects of the ever-growing epistemic distance between the processes that guide contemporary democratic procedures and the agents that partake in such procedures is nevertheless detrimental to both the agent and the process itself.

While this chapter does not go into detail of any particular instance of opacity in real-life democratic processes, the conceptual discussion of epistemic opacity in this chapter has implications for a wide range of democratic processes. These processes encompass those which

require the mediation of conventional computational infrastructure in general as well as those processes which indirectly make use of computational analytic methodology such as big data, machine learning and artificial intelligence. The processes in the former category are those in which computers are directly deployed in technology used to execute a relevant part of a democratic process, e.g. voting machines. The latter category includes other aspects of the democratic processes in which computational analytics have a direct or indirect effect on the democratic process and/or on those who take part in a democratic process. Examples of this latter processes include the kinds of analytics used to evaluate constituency and district maps but they also include the type of processes which disseminate electoral information through targeted campaigning.

Epistemic opacity

In its simplest sense, a system is deemed opaque in some form if all or some of its parts are not immediately accessible to inspection. Here, I will refer to this overly broad sense of opacity as 'general opacity'. Paul Humphreys, who contextualized the term in the literature of computer simulations, defines it as follows:

General opacity: "A process is epistemically 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, 2004; 2009)

Described this way, many things qualify as opaque to many different agents. To many of us, for example, whatever is under the hood of our cars, will be opaque. Similarly, well-established democratic procedures, such as the way the Electoral College works in the United States, remain opaque to many of those that participate in them.

A common problem related to general opacity arises from the fact that software systems often require many components. Several people and often several distinct companies also make each of those components. In this sense 'many hands' are involved in the development of software systems. The motley and compartmentalized features inherent to software development (Winsberg, 2010) bring about an analogous problem to what ethicists call the 'many hands problem' in responsibility ascription (Nissenbaum, 1994). The 'many hands problem' stipulates that attributing responsibility in a setting in which there are many agents with varying degrees and kinds of involvement is extremely difficult. Corporate, military, manufacturing, and other settings in which many agents are involved and in which the kinds and degrees of responsibilities are distributed in unequal ways are problematic for responsibility ascription. When it comes to computational methods, however, the 'many hands problem' is not only about responsibility ascription, but also about which component is causally responsible for any relevant aspect of the system's behavior. Hence, inspecting computational systems is a non-trivial epistemic challenge. If we consider that producing a single software product can take a whole research team, or a group of such teams, and that each of the individuals or groups may have varying and distributed responsibilities we can see that the 'many hands' problem has a uniquely serious effect on the way we ascribe responsibility, accountability, and/or even causality in contexts where more sophisticated computational methods play a role.

General epistemic opacity as defined above is a relative kind of opacity. Given the example above of the 'many hands problem', someone somewhere could access an existing spreadsheet with all the relevant information regarding the provenance of a given component and therefore alleviate the relative opacity vis-à-vis that particular agent. And while it may take a longer time and a great amount resources, the same can be said about each and all components of an industrially produced piece of equipment. That is, in principle, the relevant details of the opaque system in question—in this case a large scale industrial, or military project or equipment—can be known. The fact that general epistemic opacity is a relative kind of opacity also makes it less of an epistemic challenge. If you don't know how something works, for example, you can ask someone that does or can be assured that someone somewhere knows and rely on their expertise (Barberousse and Vorms, 2014). This kind of opacity also captures instances in which the same system is opaque to an epistemic agent at one time and transparent to the same epistemic agent at another time. That is, if you don't know how a system works you can, in principle, learn how it does so. Because few of us are experts at anything, or because few of us can be experts at everything, many systems/processes will be generally opaque to most of us.

Identifying a system as opaque in the manner described above offers little information concerning the different sources of opacity or the features of the system that make it so. Too many systems qualify as opaque in this manner without much information about the different ways by which such opacity occurs. To counter this overly general sense of opacity in Humphreys' definition others have suggested that knowing the specific sources of opacity (Kamisnki, 2017) and dealing with them on a case-by-case basis is the only approach to appropriately address issues of opacity and trust in technology (Pitt, 2010). Kamisnki (2017), for instance, suggests that some instances of opacity in computer simulations will arise in virtue of the social aspects of a process such as the number of people or teams that participate in a scientific project—this is another version, focused on epistemic obstacles, of the 'many hands problem' I described above. This kind of epistemic opacity, which derives from social aspects of a system, also includes instances of opacity that are due to intentional corporate and/or state secrecy, as well as the kind that arises due to technical illiteracy (Burrell, 2016). An interesting example of the kind of corporate secrecy which is of immediate social consequence are predictive algorithms deployed in a growing and diverse set of judicial procedures in the United States. These algorithms and the way they reach conclusions on things like the probability of criminal recidivism are kept opaque not only from the judges who used them for sentence or parole considerations, but also from the people whose life they affect and the lawyers navigating the judicial system on their behalf (O'Neil, 2016). The use of these technologies offers the present discussion a way to elucidate an important aspect of the nature of the challenges posed by opaque systems to democratic processes. While the consequences of using opaque predictive algorithms in the judicial system has immediate social and ethical consequences in the lives of those directly affected, the broader challenge to democratic processes as a whole is that they are ultimately closed to epistemic recourse (Kitchin, 2014): there is no way for those using them or affected by them to understand them, there is no way to navigate them and, importantly, there is no way to challenge their results. In this sense, this kind of opaque systems represent a challenge to democratic processes insofar as an informed citizenry, a transparent bureaucracy and challengeable governance are considered essential to the flourishing of any democratic system. While the court procedures of city and state judicial systems may not be directly understood as democratic processes in themselves, this example helps elucidate what makes the use of opaque

data analytics and machine learning systems in bureaucratic decision-making something to be genuinely worried about in the context of democratic institutions.

A different source of opacity from the ones discussed above emerges from the different ways in which the mathematical elements of a computational process function or malfunction together. Mathematical error, for example, is inherent in computational components. Due to discretization techniques and resource constraints, computational approximates of continuous equations rely on rounding results. Memory slots for individual digits in conventional computers, for examples, are often less than the number needed for the full representation of particularly large calculations. When each of the multiple components and processes in a computer simulation round up or down mathematical results to accommodate for these kinds of constraints, the discrepancies between what should be and what the computer produces are non-trivial. Though there are important developments in the ways in which computer scientists are able to assess and correct for this kind of error, these developments are yet to be scaled up towards a general application in software development. Rather, these developments are the kind resource-intensive proofing techniques that only high-precision coding benefits from. Knowing where and when the rounding error generates as well as the magnitude of it represents a significant epistemic challenge. Furthermore, the kind of challenge exemplified by mathematical opacity points towards a kind of epistemic opacity which may prove to be even more resistant—or in fact impervious— to inspection than the relative kind of epistemic opacity exemplified in the general definition above. This is particularly the case when a machine yields significantly different results when no changes in the conventional elements of a computational inquiry—components, code, architecture, data, parameters, etc.—can be detected (Kaminski, 2017).

Mathematical opacity also points to a kind of opacity whose origins are not rooted in limitations on the part of an agent but rather on properties and features of the system under investigation.[i] Hence, mathematical opacity is already several steps removed from the kind of opacity due to technical illiteracy mentioned above. Regardless of an agent's mathematical and technical know-how, the details of a mathematically opaque system will remain hidden.

A difference noted by Humphreys (2009) provides a possible way to sort between the instances of opacity described above. While some of the generally opaque systems are relative to a specific time and to a specific agent (agent X at time t), a subset of opaque systems will not be opaque in this relative way. Some systems will be opaque to an agent at all times or to all similar agents all the time. This can be because the nature of the agent makes it impossible to access all the epistemically relevant aspects of such a system. Humphreys calls this kind of opacity essential epistemic opacity and defines it the following way:

Essential opacity: arises "iff it is impossible, given the nature of X, for X to know all of the epistemically relevant elements of the process." (2009)

Thus, we can differentiate between those systems that are generally and relatively opaque and those that are insurmountably opaque to agents like us.[ii]

In order to better visualize the epistemic implications of a system's essential of opacity consider the following scenario. Imagine there is an error that has to be fixed in a computer such as the

direct recording electronic (DRE) systems used in the United States to vote in federal and local elections (Norden and Famighetti, 2015). Imagine an erroneous result, in a preliminary test or in an actual election, is known but the source and/or nature (hardware or software) of the error is not. When it comes to computational methods, as we briefly touched upon above, the many hands problem simply becomes more complex. The hands (teams/corporations/institutions) involved, the components necessary and the software are only some of the factors. A more troubling fact is that the inner workings of the components and algorithms in such systems aren't always accessible to other agents, users, or even those in charge of testing them (Parnas, 1990; Symons and Horner, 2014; Burrell, 2016). As I will show below, each of these components includes a large number of lines of code that make up its software and this alone can make the difference between a system that is generally opaque and one that is essentially opaque. Although some would argue that there are ways to ensure that some of the code is accessible, most of the time the practices that ensure such transparency are not available or even viable, particularly if the code in question is legacy code—code that was written by others, often long time ago—or if it is code written in a highly idiosyncratic manner. Further, as I will detail below, some efforts turn out to not be validly applicable in relevant cases. This is because computational methods have at least three very particular properties when it comes to error: the first is that, unlike any hardware technology, software is particularly sensitive to error (Parnas, et al., 1990 p.638; Corbató, 1990); second, unlike any hardware technology, the nature, source and distribution rate of its error cannot be assumed to be random (Parnas, et al.1990, p.638); and third, testing software is notably difficult (Symons and Horner, 2014; 2019).

Symons and Horner (2014) exemplify one kind of difficulty in testing software code by considering a trivially small software program consisting of 1000 lines of code. In average 1 out of 10 of those 1000 lines of code will be a command that has the form if/then/else and not just if/then. If we wanted to exhaustively check a similar size program that only had if/then commands it would be pretty straightforward. All that there is to do is to check each one of the 1000 lines of code. However, when the code includes an if/then/else command it bifurcates the possible paths that the code can take and therefore increases the lines of code to check. A straightforward program with 10 command lines has 10 lines to check. A 10-line program with one if/then/else command has 20 lines to check. A program with 1000 lines of code and an average of 1/10 commands being an if/then/else command will have enough lines of code to check to make the task, even at the fastest possible computation speed, take several times the age of the universe (Symons and Horner, 2014; Symons and Alvarado, 2016). Software systems involved in computational methods such as those required for artificial intelligence, and even some smaller subcomponents of them, have software whose number of lines of code by far exceed the number used in the example. A thorough assessment of the proper execution of each line of code then is for all immediate intents and purposes not feasible. A task that would take anywhere near the age of the universe to be completed is a task that, as a process, is opaque to many more agents than just humans and our technologies. So, this process is in effect essentially epistemically opaque.

Someone may say as an objection that if we consider a breakthrough in computing processes, like quantum parallel computing, we could significantly increase the speed at which testing software could be done. If so, exhaustively testing an average software system would be in fact possible. While this is true, it is hard to see how decreasing the time of a task from several times

the age of the universe to half or a quarter of that is genuine progress (Symons and Horner, 2014). This task remains opaque. Still, it can be argued that exhaustive testing of code lines is only one way of assessing reliability in software systems. A more useful approach is to conduct tests on random samples of a system's code. By using conventional statistical inference theory, one can randomly select samples of code and test their execution. By inference, and with the assumption of random distribution of error in the system, a reliability assessment of the samples should give us an idea of the rate of error in the whole system. If 20 sample lines out of a 100 fail to do what they are supposed to do then we can say that the system as a whole has a 20% failure rate.[iii] However, there are two main issues with this approach. Even before software systems reached the size and complexity seen today, it was well established that the nature and distribution of error in software systems is not like that of other systems (Parnas et al., 1990 p.638). This is because of two reasons. The first is that while in other engineering projects one can assume certain resilience to small errors, in software even a punctuation error can be catastrophic (1990, p.637). Second, the kinds of error that that can affect other engineering projects can be assessed by assuming that these errors are "not highly correlated" (1990 p.638). Because of the interconnectedness of software functioning and the fact that many errors are due to design (Floridi, Fresco and Primiero, 2014), errors in software cannot be considered statistically independent (Parnas et al., 1990) nor can they be considered randomly distributed (Parnas and Kwan, 1990; Horner and Symons, 2014). Thus, deploying statistical inference techniques to test samples under the assumption that the errors found will be randomly distributed is an invalid approach when it comes to software (Horner and Symons, 2019).

Further, even if this technique were valid consider the following. If the number of lines of code to be tested is of the order of magnitude postulated in the example above, what constitutes a significant sample set? Testing even 5% of the lines of code of a system with 100,000 lines of code and an average path complexity of one if/then/else bifurcation per every 10 lines would put us in the same position as with the original example. It would take ages to do. If we managed to test 1% of the lines it would be difficult to say that we now have a dependable assessment of the system's reliability. It is not clear that path complexity catastrophe applies directly to machine learning techniques. But it does apply to most of the components underlying its implementation. Big data processes are equally affected by this problem (Symons and Alvarado, 2016). Insofar as artificial intelligence algorithms are designed as symbol manipulation techniques with Boolean thresholds, they are similarly vulnerable. Similarly, if these systems rely heavily on machine learning techniques such as convolutional neural networks, then the fact remains that the components used to implement them are opaque in the ways described above.

A crucial element of reliability assessment is to understand the source, the nature, and the rate of error in a system. If our inquiry concerns the reliability of a system, as it would have been in the case of assessing computational methods deployed in the service of democratic processes, the source, nature and rate of error are relevant elements of the system. Not being able to assess the reliability of a system is enough to not trust it (Symons and Alvarado, 2019). This is particularly the case if these systems are voting machines with which voting for national elections is being done. If, as stated at the beginning of this chapter, a manageable level of transparency is essential to democratic processes and political campaigns, as part of democratic processes are socially consequential settings, then any and all computational systems used as aids to policy related issues ought to be amenable to reliability assessment. Without this, trust cannot be adequately

justified (Symons and Alvarado, 2019) and without trust democratic systems can be easily undermined.

Big Data, Machine Learning and Artificial intelligence

Despite many overlapping features, artificial intelligence, big data, and machine learning techniques can be differentiated by examining the way in which each of them is deployed and what practitioners expect to get from them (Marr, 2016). David Robinson, for example, suggests that "data science produces insights; machine learning produces predictions; and artificial intelligence produces actions." (2018). Questions in data science are often associated with explanatory and exploratory queries seeking to extract causal, inferential and sometimes even mechanistic insights from datasets (Leeks, 2015). In contrast, machine learning does not require and often cannot consider human capabilities and needs—such as explanations and/or causal orderings of a tractable size—in order to work (Burrell, 2016; Alvarado and Humphreys, 2017). There is a sense in which if machine learning processes were designed to be accessible to limited representational abilities of humans they just would not work (Burrell, 2016: 5). Unlike the processes involved in big data analytics, the techniques of machine learning are used to compute complex Bayesian probability models which focus on attaining predictions. These predictions can be about future probabilities concerning a state of affairs in a system. They can also be about the probabilities that a certain object and/or data set does or does not contain a feature identified through known or discovered properties. However, there is a sense in which all three computational methods are also often deployed simultaneously and/or are contained within one another. This is particularly the case in artificial intelligence techniques which could not be deployed without the use of big data analytics and which also often include the dynamic processes of learning algorithms in order to optimize their actionable results.

In this section I will treat all three methods as members of the same statistically-driven computational techniques in order to elucidate what sets them apart from the instances of opacity which emerge from the more conventional computational methods discussed above. When these methods are deployed in socially consequential contexts such as bureaucratic procedures or democratic processes, their epistemic challenges are of a different order. This is because the opacity at play in this context takes on many dimensions. The kind of essential epistemic opacity discussed above—related to either path complexity in software testing or mathematical opacity at the component level—is at play at the most basic level of computational systems: their code and algorithmic operation. In systems such as machine learning software the issue is exacerbated by the fact that machine learning algorithms run many iterations of the same models of analysis with slight difference in parameters. So not only are there many agents involved in the process, or many components, but also many processes. Hence, the obstacles to access the epistemically relevant features of a system are beyond their underlying algorithmic nature. In the case of artificial intelligence systems, for example sometimes thousands of substantially different models are run in order to arrive at the optimal solution for a given task. The 'many hands' problem thus becomes a 'many models' or 'many everything' problem, something that Domingos (2012) deems "the curse of dimensionality" in machine learning.

When it comes to machine learning, the number of lines of code or the number of people involved in developing the system is not the only element that contributes to its opacity. Rather, the scale of data, the number and properties of salient features that will be picked up by the

process, the alterations to the main code done by the learning aspect of the process all contribute to its complexity and opacity (Alvarado and Humphreys, 2016; Burrell 2016). This in itself constitutes an important departure from the algorithmic opacity related to testing lines of code: the algorithm itself may prove to be intractable, but also the weights and properties used by the algorithm to optimize its output will be unknown. Furthermore, the hidden statistical structures—say, for example, spuriously related data items at the deeper analytical levels—will be unreachable for anyone trying to assess whether the results are valid. Alvarado and Humphreys (2016) call this 'representational opacity'. This sort of opacity occurs when a data processing method finds a "concealed statistical structure" (2016) within a data set that cannot be interpreted by or represented to us. It arises not only from the size of the system or how complicated it is to follow, but rather from the required dynamics of its working. According to Jenna Burrell (2016) these complexities arise due to the scale at which machine learning is required to be deployed to work as intended. The many dimensions in which machine learning methods work quickly become intractable. If they are coupled with the many hands problem and the path complexity problem, we can see how the level of opacity at play in these technologies is unprecedented.

When it comes to socially consequential contexts, the validity of the results of a big data or machine learning analysis is not the only aspect of concern. The fairness and desirability of the elements considered in the analysis is also something to worry about. In this sense, even the fact that the machine will be using a discovered relation between one item an another could be questionable in itself. Predictive algorithms that take into consideration implicit or explicit racial elements for judicial procedures—whether or not they indeed show a statistically significant relation to a relevant aspect of the inquiry— are something that most advocates of fair justice systems would find undesirable (O'Neil, 2016). The main problem for the purpose of our discussion, however, is that both these kinds of deep analytic considerations and the operations necessary to discover their statistical relations will be essentially hidden from our sight.

Consider the following. When district maps are analyzed through the use of highly sophisticated big data analysis in order to optimize the outcome for candidates of a specific party, there is reason to worry. The reason, however, is not merely the fact that those using these techniques are seeking an advantage through the politically questionable practice of parsing out districts in arbitrary ways. Rather, the worry is that they are using a technique that is often not amenable to inspection and/or challenge because of the many reasons listed above. Similarly, when people are being targeted through marketing campaigns designed to exploit their cognitive vulnerabilities in order to nudge their electoral behavior (Zuboff, 2019) there is an added insult to the social injury that is rooted in the use of opaque technologies. When big data analytics, machine learning algorithms and the decisions of intelligent systems are deployed in these contexts, often neither those that use them nor those that are affected by their use fully understand the mechanisms by which they arrive at their results (O'Neil, 2016). Often, the results associated with these technologies are used to parse individuals, or groups of individuals, into actionable categories. That is, these technologies are parsing people in ways in which something can be done to them without them knowing so (Zuboff, 2019). This in itself is a significant problem to a democratic society (Murray and Scime, 2010; Cadwalladr, 2017). And yet, consider that data analytics and predictive techniques are not only used within some democratically established systems but sometimes even outside of them to assess the probabilities of democratization of seemingly undemocratic societies (Kimber, 1991). These

assessments in turn inform and possibly influence global interventions that seek, ironically, to nudge nations towards more democratic structures of government.

The common thread in the examples above, however, is that the computational methods that make such interventions possible are themselves opaque in a way that leaves the agents affected by their implications devoid of epistemic recourse. That is, given the broad discussion above of the different ways in which these technologies are opaque—inaccessible to agent understanding, inspection and/or challenge—these technologies diminish the likelihood of an informed citizenry at the helm of democratic processes. This is more reason to thread carefully in their deployment and to make sure that while they may be used for speculative and exploratory work, they are only cautiously attached to policy-making procedures. We have to come to terms with whether deploying a system whose reliability we cannot assess is something that we should do. In this sense, both what is reliable and what is trustworthy would have to be redefined in a way which omits appeals that are rooted in explicit explanatory criteria and assume the diminished—if perhaps pragmatic—epistemic stance that only predictive prowess matters when we are to justify why they are so. If this is the case, the role of the epistemic agent, us, at the core of democratic processes will be significantly reduced (Gorton, 2016).

Conclusion

This chapter does not address the many possible threats that arise when computational methods such as big data, machine learning and artificial intelligence do or do not work as intended and/or the repercussions either way.[iv] Rather, this chapter focuses on the fact that if they do or do not work as intended, the reasons why they do or fail to do so will be beyond inspection or revision because of their opacity. This fact alone should suffice as a warning going forward. This is particularly the case if our ability to create a conscious, revisable democratic environment implies the need for free, deliberate and knowledgeable choice.

Understanding the different kinds of opacity present in these computational methods is a first step towards transparency. If the opacity in question is insurmountable, then understanding so provides at least a way to consciously allocate resources where they may make a difference. It will also provide a reason to think twice about the contexts and the ways in which we deploy and trust these technologies going forward.

References

Alvarado, R. and Humphreys, P., 2017. Big Data, Thick Mediation, and Representational Opacity. *New Literary History*, *48*(4), pp.729-749.

Amoore, L., 2014. Security and the incalculable. *Security Dialogue*, *45*(5), pp.423-439.

Boyd, Danah, and Kate Crawford. "Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon." *Information, communication & society* 15, no. 5 (2012): 662-679.

Burrell, J., 2016. How the machine ‘thinks’: Understanding opacity in machine learning algorithms. *Big Data & Society*, *3*(1), p.2053951715622512.

Cadwalladr, C., 2017. The great British Brexit robbery: how our democracy was hijacked. *The Guardian*, *20*.

Corbató, Fernando J. "On building systems that will fail." In *ACM Turing award lectures*, p. 1990. ACM, 2007.

Domingos, P., 2012. A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), pp.78-87.

Floridi, Luciano, Nir Fresco, and Giuseppe Primiero. "On malfunctioning software." *Synthese* 192.4 (2015): 1199-1220.

Gorton, W.A., 2016. Manipulating Citizens: How Political Campaigns’ Use of Behavioral Social Science Harms Democracy. *New Political Science*, *38*(1), pp.61-80.

Horner, Jack, and John Symons. "Reply to Angius and Primiero on software intensive science." *Philosophy & Technology* 27, no. 3 (2014): 491-494.

Humphreys, Paul. "The philosophical novelty of computer simulation methods." *Synthese* 169, no. 3 (2009): 615-626.

Kaminski, Andreas. "Der Erfolg der Modellierung und das Ende der Modelle. Epistemische Opazität in der Computersimulation." (2017).

Kimber, Richard. "Artificial intelligence and the study of democracy." *Social Science Computer Review* 9, no. 3 (1991): 381-398.

Kitchin, Rob. *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage, 2014.

Marr, Bernard. "What Is the Difference Between Artificial Intelligence and Machine Learning." *Forbes, December* 6 (2016).

Murray, G.R. and Scime, A., 2010. Microtargeting and electorate segmentation: data mining the American National Election Studies. *Journal of Political Marketing*, *9*(3), pp.143-166.

Nissenbaum, Helen. "Computing and accountability." *Communications of the ACM* 37, no. 1 (1994): 72-81.

O'Neil, Cathy. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books, 2016.

Parnas, David L., A. John Van Schouwen, and Shu Po Kwan. "Evaluation of safety-critical software." *Communications of the ACM* 33, no. 6 (1990): 636-648.

Pitt, Joseph C. "It's not about technology." *Knowledge, Technology & Policy* 23, no. 3-4 (2010): 445-454.

Symons, John, and Ramón Alvarado. "Can we trust Big Data? Applying philosophy of science to software." *Big Data & Society* 3, no. 2 (2016): 2053951716664747.

Symons, John, and Ramón Alvarado. "Epistemic entitlements and the practice of computer simulation." *Minds and Machines* 29, no. 1 (2019): 37-60.

Symons John F. and Jack Horner " Software Error as a Limit to Inquiry for Finite Agents: Challenges for the Post-human Scientist in Powers, T. (ed.) Philosophy and Computing: Essays in Epistemology, Philosophy of Mind, Logic, and Ethics. Springer pp. 2017

Winsberg, Eric. *Science in the age of computer simulation*. University of Chicago Press, 2010.

Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books.

[i] This is in fact a very important distinction, albeit one for which I do not have space to elaborate in this particular chapter. Suffice it to say that in the context of explanations of epistemic opacity some will point towards the limitations on the part of the agent and some will point to features of a system. As it will become clear as we move forward, what both general and essential opacity have in common is that they are agent-based accounts of opacity. Both place the source of opacity within the agent. Furthermore, some other systems/processes will be opaque due to limitations that almost any other agent would have, even agents significantly superior to us. Inevitably, while the distinction between general and essential opacity will capture many instances, it will fail to capture instances of opacity that a) do not respond to or b) do not emerge from the limitations or abilities of agents. I call these instances of opacity agent-neutral and agent-independent respectively. Similarly, it would be incorrect to assert that the opacity in agent-neutral cases is due to the nature of X when X stands for an individual/specific kind of agent if the instance of opacity arises in virtue of a property shared by all finite epistemic agents.

[iii] The probability assessment in actual software testing are a lot more sophisticated and intricate than the example shows, but for all intents and purposes, they function by the same statistical principles whose assumption is being challenged here.

[iv] For a thorough overview of these threats see boyd and Crawford (2012), Amoore (2014) O'Neill (2016), and more recently Zuboff (2019).