Summarize

Chapter 11. The Human Side of Machine Learning

Throughout this book, we've covered many technical aspects of designing an ML system. However, ML systems aren't just technical. They involve business decision makers, users, and, of course, developers of the systems. We've discussed stakeholders and their objectives in Chapters 1 and 2. In this chapter, we'll discuss how users and developers of ML systems might interact with these systems.

We'll first consider how user experience might be altered and affected due to the probabilistic nature of ML models. We'll continue to discuss organizational structure to allow different developers of the same ML system to work together effectively. We'll end the chapter with how ML systems can affect the society as a whole in the section "Responsible AI".

User Experience

We've discussed at length how ML systems behave differently from traditional software systems. First, ML systems are probabilistic instead of deterministic. Usually, if you run the same software on the same input twice at different times, you can expect the same result. However, if you run the same ML system twice at different times on the exact same input, you might get different results.[1] Second, due to this probabilistic nature, ML systems' predictions are mostly correct, and the hard part is we usually don't know for what inputs the system will be correct! Third, ML systems can also be large and might take an unexpectedly long time to produce a prediction.

These differences mean that ML systems can affect user experience differently, especially for users that have so far been used to traditional soft-

ware. Due to the relatively new usage of ML in the real world, how ML systems affect user experience is still not well studied. In this section, we'll discuss three challenges that ML systems pose to good user experience and how to address them.

Ensuring User Experience Consistency

When using an app or a website, users expect a certain level of consistency. For example, I'm used to Chrome having their "minimize" button on the top left corner on my MacBook. If Chrome moved this button to the right, I'd be confused, even frustrated.

ML predictions are probabilistic and inconsistent, which means that predictions generated for one user today might be different from what will be generated for the same user the next day, depending on the context of the predictions. For tasks that want to leverage ML to improve users' experience, the inconsistency in ML predictions can be a hindrance.

To make this concrete, consider a case study published by Booking.com in 2020. When you book accommodations on Booking.com, there are about 200 filters you can use to specify your preferences, such as "breakfast included," "pet friendly," and "non-smoking rooms." There are so many filters that it takes time for users to find the filters that they want. The applied ML team at Booking.com wanted to use ML to automatically suggest filters that a user might want, based on the filters they've used in a given browsing session.

The challenge they encountered is that if their ML model kept suggesting different filters each time, users could get confused, especially if they couldn't find a filter that they had already applied before. The team resolved this challenge by creating a rule to specify the conditions in which the system must return the same filter recommendations (e.g., when the user has applied a filter) and the conditions in which the system can return new recommendations (e.g., when the user changes their destination). This is known as the consistency–accuracy trade-off, since the recommendations deemed most accurate by the system might not be the recommendations that can provide user consistency.

Combatting "Mostly Correct" Predictions

In the previous section, we talked about the importance of ensuring the consistency of a model's predictions. In this section, we'll talk about how, in some cases, we want less consistency and more diversity in a model's predictions.

Since 2018, the large language model GPT and its successors, GPT-2 and GPT-3, have been taking the world by storm. An advantage of these large language models is that they're able to generate predictions for a wide range of tasks with little to no task-specific training data required. For example, you can use the requirements for a web page as an input to the model, and it'll output the React code needed to create that web page, as shown in Figure 11-1.

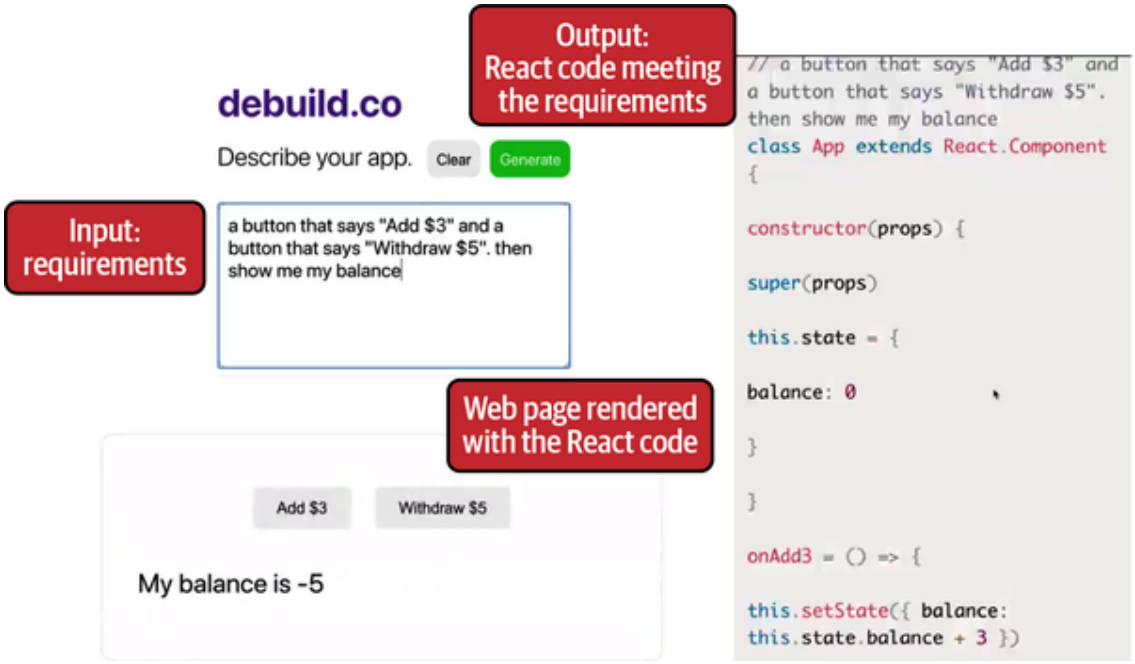

Figure 11-1. GPT-3 can help you write code for your website. Source: Adapted from screenshots of a video by Sharif Shameem

However, a drawback of these models is that these predictions are not always correct, and it's very expensive to fine-tune them on task-specific data to improve their predictions. These mostly correct predictions can be useful for users who can easily correct them. For example, in the case of customer support, for each customer request, ML systems can produce mostly correct responses and the human operators can quickly edit those responses. This can speed up the response compared to having to write the response from scratch.

However, these mostly correct predictions won't be very useful if users don't know how to or can't correct the responses. Consider the same task of using a language model to generate React code for a web page. The generated code might not work, or if it does, it might not render to a web page that meets the specified requirements. A React engineer might be able to fix this code quickly, but many users of this application might not know React. And this application might attract a lot of users who don't know React—that's why they needed this app in the first place!

To overcome this, an approach is to show users multiple resulting predictions for the same input to increase the chance of at least one of them being correct. These predictions should be rendered in a way that even non-expert users can evaluate them. In this case, given a set of requirements input by users, you can have the model produce multiple snippets of React code. The code snippets are rendered into visual web pages so that nonengineering users can evaluate which one is the best for them.

This approach is very common and is sometimes called "human-in-the-loop" AI, as it involves humans to pick the best predictions or to improve on the machine-generated predictions. For readers interested in human-in-the-loop AI, I'd highly recommend Jessy Lin's "Rethinking Human-AI Interaction".

Smooth Failing

We've talked at length about the effect of an ML model's inference latency on user experience in the section "Computational priorities". We've also discussed how to compress models and optimize them for faster inference speed in the section "Model Compression". However, normally fast models might still take time with certain queries. This can happen especially with models that deal with sequential data like language models or time-series models—e.g., the model takes longer to process long series than shorter series. What should we do with the queries where models take too long to respond?

Some companies that I've worked with use a backup system that is less optimal than the main system but is guaranteed to generate predictions quickly. These systems can be heuristics or simple models. They can even be cached precomputed predictions. This means that you might have a rule that specifies: if the main model takes longer than X milliseconds to

generate predictions, use the backup model instead. Some companies, instead of having this simple rule, have another model to predict how long it'll take the main model to generate predictions for a given query, and route that prediction to either the main model or the backup model accordingly. Of course, this added model might also add extra inference latency to your system.

This is related to the speed–accuracy trade-off: a model might have worse performance than another model but can do inference much faster. This less-optimal but fast model might give users worse predictions but might still be preferred in situations where latency is crucial. Many companies have to choose one model over another, but with a backup system, you can do both.

Team Structure

An ML project involves not only data scientists and ML engineers, but also other types of engineers such as DevOps engineers and platform engineers as well as nondeveloper stakeholders like subject matter experts (SMEs). Given a diverse set of stakeholders, the question is what is the optimal structure when organizing ML teams. We'll focus on two aspects: cross-functional teams collaboration and the much debated role of an end-to-end data scientist.

Cross-functional Teams Collaboration

SMEs (doctors, lawyers, bankers, farmers, stylists, etc.) are often overlooked in the design of ML systems, but many ML systems wouldn't work without subject matter expertise. They're not only users but also developers of ML systems.

Most people only think of subject matter expertise during the data labeling phase—e.g., you'd need trained professionals to label whether a CT scan of a lung shows signs of cancer. However, as training ML models becomes an ongoing process in production, labeling and relabeling might also become an ongoing process spanning the entire project lifecycle. An ML system would benefit a lot to have SMEs involved in the rest of the lifecycle, such as problem formulation, feature engineering, error analy-

sis, model evaluation, reranking predictions, and user interface: how to best present results to users and/or to other parts of the system.

There are many challenges that arise from having multiple different profiles working on a project. For example, how do you explain ML algorithms' limitations and capacities to SMEs who might not have engineering or statistical backgrounds? To build an ML system, we want everything to be versioned, but how do you translate domain expertise (e.g., if there's a small dot in this region between X and Y then it might be a sign of cancer) into code and version that?

Good luck trying to get your doctor to use Git.

It's important to involve SMEs early on in the project planning phase and empower them to make contributions without having to burden engineers to give them access. For example, to help SMEs get more involved in the development of ML systems, many companies are building no-code/low-code platforms that allow people to make changes without writing code. Most of the no-code ML solutions for SMEs are currently at the labeling, quality assurance, and feedback stages, but more platforms are being developed to aid in other critical junctions such as dataset creation and views for investigating issues that require SME input.

End-to-End Data Scientists

Through this book, I hope I've convinced you that ML production is not just an ML problem but also an infrastructure problem. To do MLOps, we need not only ML expertise but also Ops (operational) expertise, especially around deployment, containerization, job orchestration, and workflow management.

To be able to bring all these areas of expertise into an ML project, companies tend to follow one of the two following approaches: have a separate team to manage all the Ops aspects or include data scientists on the team and have them own the entire process.

Let's take a closer look at how each of these approaches works in practice.

Approach 1: Have a separate team to manage production

In this approach, the data science/ML team develops models in the dev environment. Then a separate team, usually the Ops/platform/ML engineering team, productionizes the models in prod. This approach makes hiring easier as it's easier to hire people with one set of skills instead of people with multiple sets of skills. It might also make life easier for each person involved, as they only have to focus on one concern (e.g., developing models or deploying models). However, this approach has many drawbacks:

Communication and coordination overhead

A team can become blockers for other teams. According to Frederick P. Brooks, "What one programmer can do in one month, two programmers can do in two months."

Debugging challenges

When something fails, you don't know whether your team's code or some other team's code might have caused it. It might not have been because of your company's code at all. You need cooperation from multiple teams to figure out what's wrong.

Finger-pointing

Even when you've figured out what went wrong, each team might think it's another team's responsibility to fix it.

Narrow context

No one has visibility into the entire process to optimize/improve it. For example, the platform team has ideas on how to improve the infrastructure but they can only act on requests from data scientists, but data scientists don't have to deal with infrastructure so they have less incentives to proactively make changes to it.

Approach 2: Data scientists own the entire process

In this approach, the data science team also has to worry about productionizing models. Data scientists become grumpy unicorns, expected to know everything about the process, and they might end up writing more boilerplate code than data science.

About a year ago, I tweeted about a set of skills I thought was important to become an ML engineer or data scientist, as shown in Figure 11-2. The list covers almost every part of the workflow: querying data, modeling, distributed training, and setting up endpoints. It even includes tools like Kubernetes and Airflow.

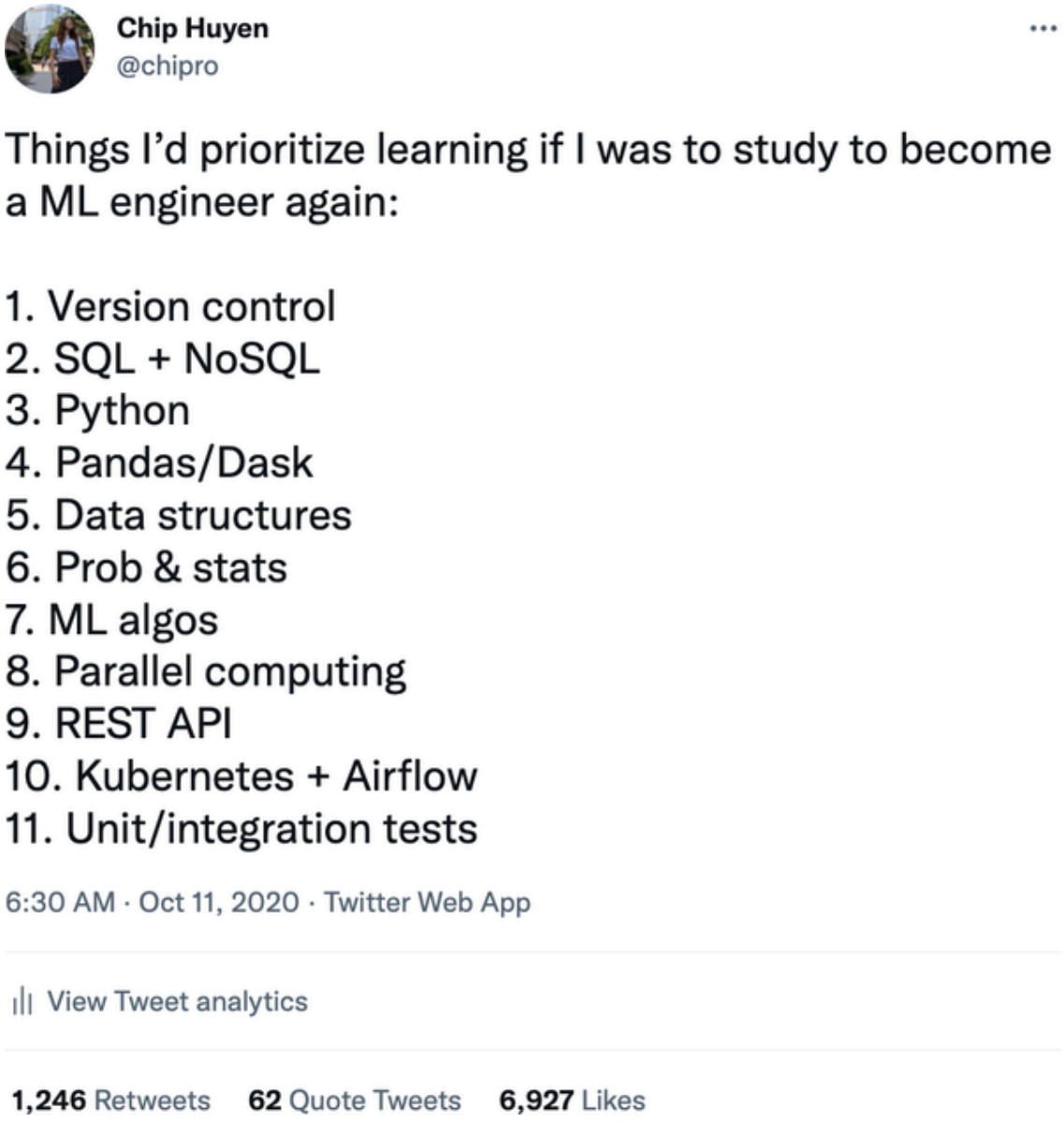

Figure 11-2. I used to think that a data scientist would need to know all these things

The tweet seems to resonate with my audience. Eugene Yan also wrote about how "data scientists should be more end-to-end."[2] Eric Colson, Stitch Fix's chief algorithms officer (who previously was also VP data science and engineering at Netflix), wrote a post on "the power of the full-stack data science generalist and the perils of division of labor through function."[3]

When I wrote that tweet, I believed that Kubernetes was essential to the ML workflow. This sentiment came from the frustration at my own job—

my life as an ML engineer would've been much easier if I was more fluent with K8s.

However, as I learned more about low-level infrastructure, I realized how unreasonable it is to expect data scientists to know about it. Infrastructure requires a very different set of skills from data science. In theory, you can learn both sets of skills. In practice, the more time you spend on one means the less time you spend on the other. I love Erik Bernhardsson's analogy that expecting data scientists to know about infrastructure is like expecting app developers to know about how Linux kernels work.[4] I joined an ML company because I wanted to spend more time with data, not with spinning up AWS instances, writing Dockerfiles, scheduling/scaling clusters, or debugging YAML configuration files.

For data scientists to own the entire process, we need good tools. In other words, we need good infrastructure.

What if we have an abstraction to allow data scientists to own the process end-to-end without having to worry about infrastructure?

What if I can just tell this tool, "Here's where I store my data (S3), here are the steps to run my code (featurizing, modeling), here's where my code should run (EC2 instances, serverless stuff like AWS Batch, Function, etc.), here's what my code needs to run at each step (dependencies)," and then this tool manages all the infrastructure stuff for me?

According to both Stitch Fix and Netflix, the success of a full-stack data scientist relies on the tools they have. They need tools that "abstract the data scientists from the complexities of containerization, distributed processing, automatic failover, and other advanced computer science concepts."[5]

In Netflix's model, the specialists—people who originally owned a part of the project—first create tools that automate their parts, as shown in Figure 11-3. Data scientists can leverage these tools to own their projects end-to-end.

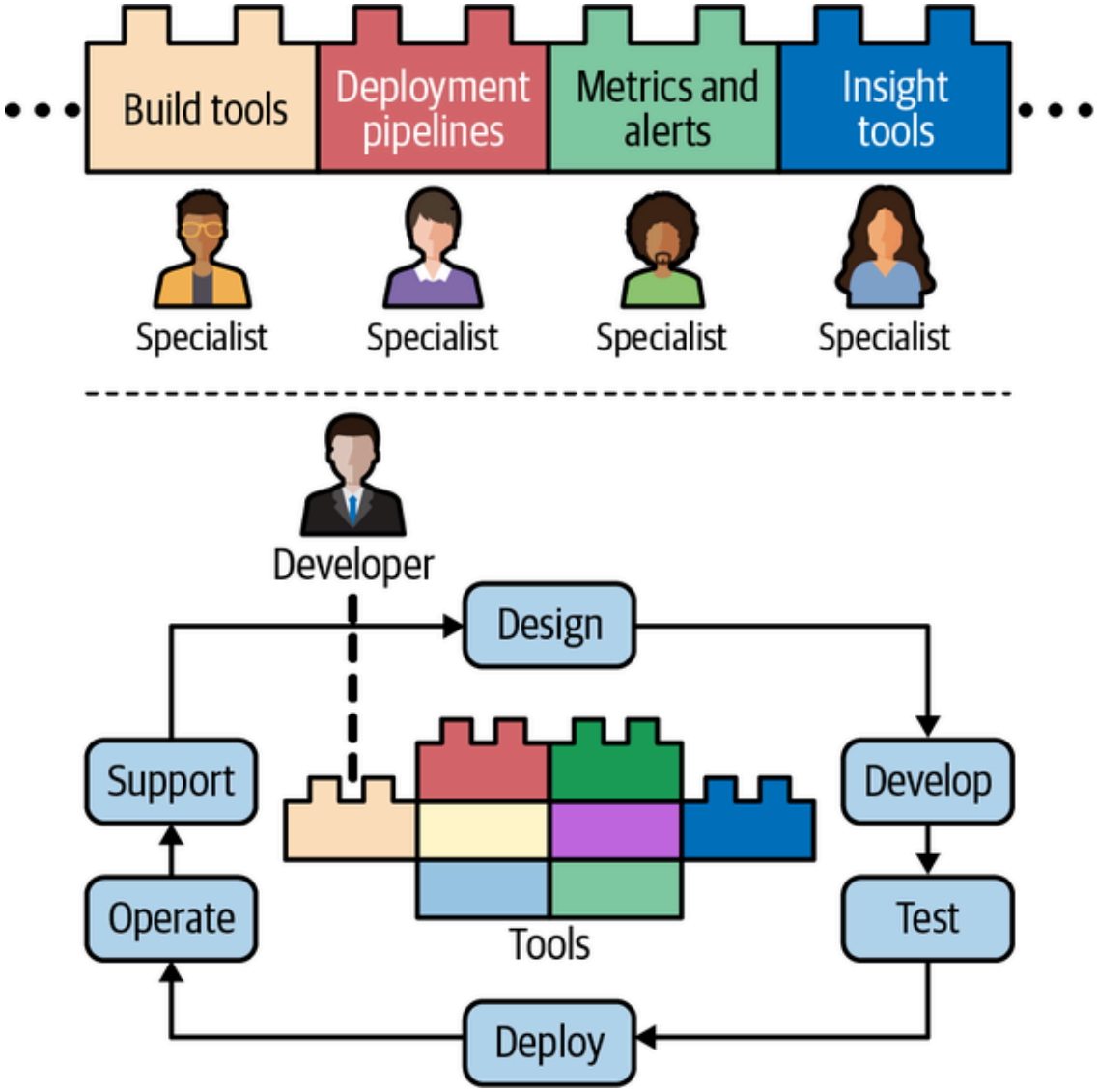

Figure 11-3. Full-cycle developers at Netflix. Source: Adapted from an image by Netflix[6]

We've talked about how ML systems might affect user experience and how organizational structure might influence productivity of ML projects. In the second half of this chapter, we'll focus on an even more crucial consideration: how ML systems might affect society and what ML system developers should do to ensure that the systems they develop do more good than harm.

Responsible AI

This section was written with generous contributions from Abhishek Gupta, founder and principal researcher at the Montreal AI Ethics Institute. His work focuses on applied technical and policy measures to build ethical, safe, and inclusive AI systems.

NOTE

The question of how to make intelligent systems responsible is relevant not only to ML systems but also general artificial intelligence (AI) systems. AI is a broader term that includes ML. Therefore, in this section, we use AI instead of ML.

Responsible AI is the practice of designing, developing, and deploying AI systems with good intention and sufficient awareness to empower users, to engender trust, and to ensure fair and positive impact to society. It consists of areas like fairness, privacy, transparency, and accountability.

These terms are no longer just philosophical musings, but serious considerations for both policy makers and everyday practitioners. Given ML is being deployed into almost every aspect of our lives, failing to make our ML systems fair and ethical can lead to catastrophic consequences, as outlined in the book *Weapons of Math Destruction* (Cathy O'Neil, Crown Books, 2016), and through other case studies mentioned throughout this book.

As developers of ML systems, you have the responsibility not only to think about how your systems will impact users and society at large, but also to help all stakeholders better realize their responsibilities toward the users by concretely implementing ethics, safety, and inclusivity into your ML systems. This section is a brief introduction to what can happen when insufficient efforts are spent to make ML systems responsible. We'll start with two case studies of quite unfortunate and public failures of ML. We will then propose a preliminary framework for data scientists and ML engineers to select the tools and guidelines that best help with making your ML systems responsible.

Disclaimer: Responsible AI is a complex topic with growing literature that deserves its own coverage and can easily span multiple books. This section is far from an exhaustive guide. We only aim to give ML developers an overview to effectively navigate the developments in this field. Those interested in further reading are highly recommended to check out the following resources:

- NIST Special Publication 1270: Towards a Standard for Identifying and Managing Bias in Artificial Intelligence

- ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT) publications
- Trustworthy ML's list of recommended resources and fundamental papers for researchers and practitioners who want to learn more about trustworthy ML
- Sara Hooker's awesome slide deck on fairness, security, and governance in machine learning (2022)
- Timnit Gebru and Emily Denton's tutorials on fairness, accountability, transparency, and ethics (2020)

Irresponsible AI: Case Studies

We'll start this section off by looking at two failures of AI systems that led to severe harm for not only the users of these systems but also to the organizations who developed the systems. We'll trace some of the places where the organizations went wrong and what the practitioners could have done to potentially anticipate these points of failure. These highlights will serve as background as we dive into the engineering framework for responsible AI.

There are other interesting examples of "AI incidents" logged at the AI Incident Database. Keep in mind that while the following two examples and the ones logged at AI Incident Database are the ones that caught attention, there are many more instances of irresponsible AI that happen silently.

Case study I: Automated grader's biases

In the summer of 2020, the United Kingdom canceled A levels, the high-stakes exams that determine college placement, due to the COVID-19 pandemic. Ofqual, the regulatory body for education and examinations in the UK, sanctioned the use of an automated system to assign final A-level grades to students—without them taking the test. According to Jones and Safak from Ada Lovelace Institute, "Awarding students' grades based on teacher assessment was originally rejected by Ofqual on the grounds of unfairness between schools, incomparability across generations and devaluing of results because of grade inflation. The fairer option, Ofqual surmised, was to combine previous attainment data and teacher assessment to assign grades, using a particular statistical model—an 'algorithm.'"[7]

The results published by this algorithm, however, turned out to be unjust and untrustworthy. They quickly led to public outcries to get rid of it, with hundreds of students chanting in protest.[8]

What caused the public outcries? The first glance seems to point at the algorithm's poor performance. Ofqual stated that their model, tested on 2019 data, had about 60% average accuracy across A-level subjects.[9] This means that they expected 40% of the grades assigned by this model to be different from the students' actual grades.

While the model's accuracy seems low, Ofqual defended their algorithm as being broadly comparable to the accuracy of human graders. When comparing an examiner's grades with those made by a senior examiner, the agreement is also around 60%.[10] The accuracy by both human examiners and the algorithm exposes the underlying uncertainty in assessing students at a single point in time,[11] further fueling the frustration of the public.

If you've read this book thus far, you know that coarse-grained accuracy alone is nowhere close to being sufficient to evaluate a model's performance, especially for a model whose performance can influence the future of so many students. A closer look into this algorithm reveals at least three major failures along the process of designing and developing this automated grading system:

- Failure to set the right objective
- Failure to perform fine-grained evaluation to discover potential biases
- Failure to make the model transparent

We'll go into detail about each of these failures. Keep in mind that even if these failures are addressed, the public might still be upset with the auto-grading system.

Failure 1: Setting the wrong objective

We discussed in Chapter 2 how the objective of an ML project will affect the resulting ML system's performance. When developing an automated system to grade students, you would've thought that the objective of this system would be "grading accuracy for students."

However, the objective that Ofqual seemingly chose to optimize was "maintaining standards" across schools—fitting the model's predicted grades to historical grade distributions from each school. For example, if school A had historically outperformed school B in the past, Ofqual wanted an algorithm that, on average, also gives students from school A higher grades than students from school B. Ofqual prioritized fairness between schools over fairness between students—they preferred a model that gets school-level results right over another model that gets each individual's grades right.

Due to this objective, the model disproportionately downgraded high-performing cohorts from historically low-performing schools. A students from classes where students had historically received straight Ds were downgraded to Bs and Cs.[12]

Ofqual failed to take into account the fact that schools with more resources tend to outperform schools with fewer resources. By prioritizing schools' historical performance over students' current performance, this auto-grader punished students from low resource schools, which tend to have more students from underprivileged backgrounds.

Failure 2: Insufficient fine-grained model evaluation to discover biases

Bias against students from historically low-performing schools is only one of the many biases discovered about this model after the results were brought to the public. The automated grading system took into account teachers' assessments as inputs but failed to address teachers' inconsistency in evaluation across demographic groups. It also "does not take into consideration the impact of multiple disadvantages for some protected groups [under the] 2010 Equalities Act, who will be double/triple disadvantaged by low teacher expectations, [and] racial discrimination that is endemic in some schools."[13]

Because the model took into account each school's historical performance, Ofqual acknowledged that their model didn't have enough data for small schools. For these schools, instead of using this algorithm to assign final grades, they only used teacher-assessed grades. In practice, this led to "better grades for private school students who tend to have smaller classes."[14]

It might have been possible to discover these biases through the public release of the model's predicted grades with fine-grained evaluation to understand their model's performance for different slices of data—e.g., evaluating the model's accuracy for schools of different sizes and for students from different backgrounds.

Failure 3: Lack of transparency

Transparency is the first step in building trust in systems, yet Ofqual failed to make important aspects of their auto-grader public before it was too late. For example, they didn't let the public know that the objective of their system was to maintain fairness between schools until the day the grades were published. The public, therefore, couldn't express their concern over this objective as the model was being developed.

Further, Ofqual didn't let teachers know how their assessments would be used by the auto-grader until after the assessments and student ranking had been submitted. Ofqual's rationale was to avoid teachers attempting to alter their assessments to influence the model's predictions. Ofqual chose not to release the exact model being used until results day to ensure that everyone would find out their results at the same time.

These considerations came from good intention; however, Ofqual's decision to keep their model development in the dark meant that their system didn't get sufficient independent, external scrutiny. Any system that operates on the trust of the public should be reviewable by independent experts trusted by the public. The Royal Statistical Society (RSS), in their inquiry into the development of this auto-grader, expressed concerns over the composition of the "technical advisory group" that Ofqual put together to evaluate the model. RSS indicated that "without a stronger procedural basis to ensure statistical rigor, and greater transparency about the issues that Ofqual is examining,"[15] the legitimacy of Ofqual's statistical model is questionable.

This case study shows the importance of transparency when building a model that can make a direct impact on the lives of so many people, and what the consequences can be for failing to disclose important aspects of your model at the right time. It also shows the importance of choosing the right objective to optimize, as the wrong objective (e.g., prioritizing fair-

ness among schools) can not only lead you to choose a model that underperforms for the right objective, but also perpetuate biases.

It also exemplifies the currently mucky boundary between what should be automated by algorithms and what should not. There must be people in the UK government who think it's OK for A-level grading to be automated by algorithms, but it's also possible to argue that due to the potential for catastrophic consequences of the A-level grading, it should never have been automated in the first place. Until there is a clearer boundary, there will be more cases of misusing AI algorithms. A clearer boundary can only be achieved with more investments in time and resources as well as serious considerations from AI developers, the public, and the authorities.

Case study II: The danger of "anonymized" data

This case study is interesting to me because here, the algorithm is not an explicit culprit. Rather it's how the interface and collection of data is designed that allows the leakage of sensitive data. Since the development of ML systems relies heavily on the quality of data, it's important for user data to be collected. The research community needs access to high-quality datasets to develop new techniques. Practitioners and companies require access to data to discover new use cases and develop new AI-powered products.

However, collecting and sharing datasets might violate the privacy and security of the users whose data is part of these datasets. To protect users, there have been calls for anonymization of personally identifiable information (PII). According to the US Department of Labor, PII is defined as "any representation of information that permits the identity of an individual to whom the information applies to be reasonably inferred by either direct or indirect means" such as name, address, or telephone number.[16]

However, anonymization may not be a sufficient guarantee for preventing data misuse and erosion of privacy expectations. In 2018, online fitness tracker Strava published a heatmap showing the paths it records of its users around the world as they exercise, e.g., running, jogging, or swimming. The heatmap was aggregated from one billion activities recorded between 2015 and September 2017, covering 27 billion kilome-

ters of distance. Strava stated that the data used had been anonymized, and "excludes activities that have been marked as private and user-defined privacy zones."[17]

Since Strava was used by military personnel, their public data, despite anonymization, allowed people to discover patterns that expose activities of US military bases overseas, including the "forward operating bases in Afghanistan, Turkish military patrols in Syria, and a possible guard patrol in the Russian operating area of Syria."[18] An example of these discriminating patterns is shown in Figure 11-4. Some analysts even suggested that the data could reveal the names and heart rates of individual Strava users.[19]

So where did the anonymization go wrong? First, Strava's default privacy setting was "opt-out," meaning that it requires users to manually opt out if they don't want their data to be collected. However, users have pointed out that these privacy settings aren't always clear and can cause surprises to users.[20] Some of the privacy settings can only be changed through the Strava website rather than in its mobile app. This shows the importance of educating users about your privacy settings. Better, data opt-in (data collecting isn't by default), not opt-out, should be the default.

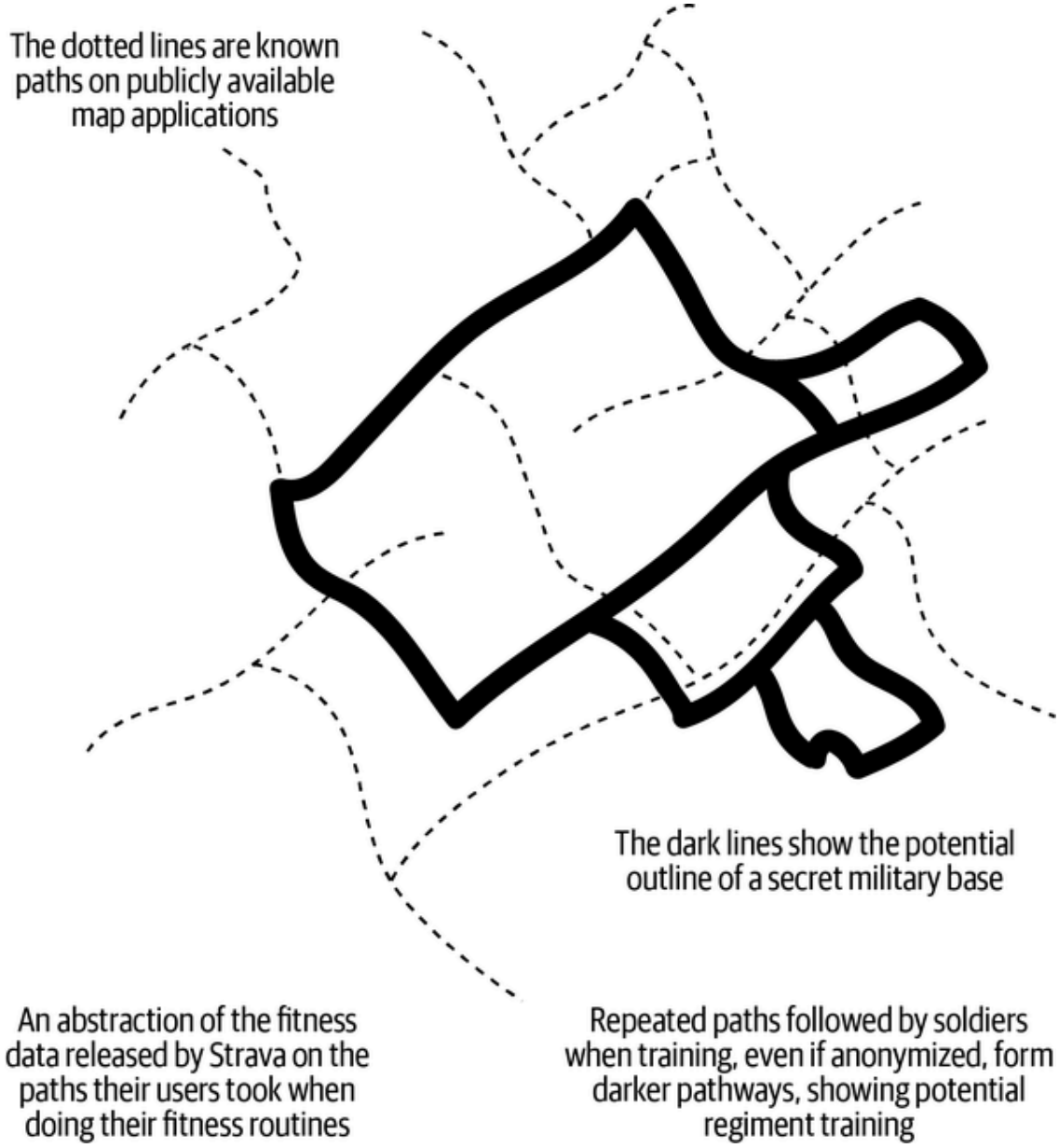

Figure 11-4. Image created based on analysis done by BBC News[21]

When this issue with the Strava heatmap became public, some of the responsibilities were shifted toward users: e.g., how military personnel shouldn't use non-military-issue devices with GPS tracking and how location services should be turned off.[22]

However, privacy settings and users' choices only address the problem at a surface level. The underlying problem is that the devices we use today are constantly collecting and reporting data on us. This data has to be moved and stored somewhere, creating opportunities for it to be intercepted and misused. The data that Strava has is small compared to much more widely used applications like Amazon, Facebook, Google, etc. Strava's blunder might have exposed military bases' activities, but other privacy failures might cause even more dangers not only to individuals but also to society at large.

Collecting and sharing data is essential for the development of data-driven technologies like AI. However, this case study shows the hidden danger of collecting and sharing data, even when data is supposedly anonymized and was released with good intention. Developers of applications that gather user data must understand that their users might not have the technical know-how and privacy awareness to choose the right privacy settings for themselves, and so developers must proactively work to make the right settings the default, even at the cost of gathering less data.

A Framework for Responsible AI

In this section, we will lay down the foundations for you, as an ML practitioner, to audit model behavior and set out guidelines that best help you meet the needs of your projects. This framework is not sufficient for every use case. There are certain applications where the use of AI might altogether be inappropriate or unethical (e.g., criminal sentencing decisions, predictive policing), regardless of which framework you follow.

Discover sources for model biases

As someone who has been following the discussions around ML systems design, you know that biases can creep in your system through the entire workflow. Your first step is to discover how these biases can creep in. The following are some examples of the sources of data, but keep in mind that this list is far from being exhaustive. One of the reasons why biases are so hard to combat is that biases can come from any step during a project lifecycle.

Training data

Is the data used for developing your model representative of the data your model will handle in the real world? If not, your model might be biased against the groups of users with less data represented in the training data.

Labeling

If you use human annotators to label your data, how do you measure the quality of these labels? How do you ensure that annotators follow standard guidelines instead of relying on subjective experi-

ence to label your data? The more annotators have to rely on their subjective experience, the more room for human biases.

Feature engineering

Does your model use any feature that contains sensitive information? Does your model cause a disparate impact on a subgroup of people? Disparate impact occurs "when a selection process has widely different outcomes for different groups, even as it appears to be neutral."[23] This can happen when a model's decision relies on information correlated with legally protected classes (e.g., ethnicity, gender, religious practice) even when this information isn't used in training the model directly. For example, a hiring process can cause disparate impact by race if it leverages variables correlated with race such as zip code and high school diplomas. To mitigate this potential disparate impact, you might want to use disparate impact remover techniques proposed by Feldman et al. in "Certifying and Removing Disparate Impact" or to use the function `DisparateImpactRemover` implemented by AI Fairness 360 (AIF360). You can also identify hidden bias in variables (which can then be removed from the training set) using the Infogram method, implemented in H2O.

Model's objective

Are you optimizing your model using an objective that enables fairness to all users? For example, are you prioritizing your model's performance on all users, which skews your model toward the majority group of users?

Evaluation

Are you performing adequate, fine-grained evaluation to understand your model's performance on different groups of users? This is covered in the section "Slice-based evaluation". Fair, adequate evaluation depends on the existence of fair, adequate evaluation data.

Understand the limitations of the data-driven approach

ML is a data-driven approach to solving problems. However, it's important to understand that data isn't enough. Data concerns people in the

real world, with socioeconomic and cultural aspects to consider. We need to gain a better understanding of the blind spots caused by too much reliance on data. This often means crossing over disciplinary and functional boundaries, both within and outside the organization, so that we can account for the lived experiences of those who will be impacted by the systems that we build.

As an example, to build an equitable automated grading system, it's essential to work with domain experts to understand the demographic distribution of the student population and how socioeconomic factors get reflected in the historical performance data.

Understand the trade-offs between different desiderata

When building an ML system, there are different properties you might want this system to have. For example, you might want your system to have low inference latency, which could be obtained by model compression techniques like pruning. You might also want your model to have high predictive accuracy, which could be achieved by adding more data. You might also want your model to be fair and transparent, which could require the model and the data used to develop this model to be made accessible for public scrutiny.

Often, ML literature makes the unrealistic assumption that optimizing for one property, like model accuracy, holds all others static. People might discuss techniques to improve a model's fairness with the assumption that this model's accuracy or latency will remain the same. However, in reality, improving one property can cause other properties to degrade. Here are two examples of these trade-offs:

Privacy versus accuracy trade-off

> According to Wikipedia, differential privacy is "a system for publicly sharing information about a dataset by describing the patterns of groups within the dataset while withholding information about individuals in the dataset. The idea behind differential privacy is that if the effect of making an arbitrary single substitution in the database is small enough, the query result cannot be used to infer much about any single individual, and therefore provides privacy."[24]

Differential privacy is a popular technique used on training data for ML models. The trade-off here is that the higher the level of privacy that differential privacy can provide, the lower the model's accuracy. However, this accuracy reduction isn't equal for all samples. As pointed out by Bagdasaryan and Shmatikov (2019), "the accuracy of differential privacy models drops much more for the underrepresented classes and subgroups."[25]

Compactness versus fairness trade-off

In Chapter 7, we talked at length about various techniques for model compression such as pruning and quantization. We learned that it's possible to reduce a model's size significantly with minimal cost of accuracy, e.g., reducing a model's parameter count by 90% with minimal accuracy cost.

The minimal accuracy cost is indeed minimal if it's spread uniformly across all classes, but what if the cost is concentrated in only a few classes? In their 2019 paper, "What Do Compressed Deep Neural Networks Forget?," Hooker et al. found that "models with radically different numbers of weights have comparable top-line performance metrics but diverge considerably in behavior on a narrow subset of the dataset."[26] For example, they found that compression techniques amplify algorithmic harm when the protected feature (e.g., sex, race, disability) is in the long tail of the distribution. This means that compression disproportionately impacts underrepresented features.[27]

Another important finding from their work is that while all compression techniques they evaluated have a nonuniform impact, not all techniques have the same level of disparate impact. Pruning incurs a far higher disparate impact than is observed for the quantization techniques that they evaluated.[28]

Similar trade-offs continue to be discovered. It's important to be aware of these trade-offs so that we can make informed design decisions for our ML systems. If you are working with a system that is compressed or differentially private, allocating more resources to auditing model behavior is recommended to avoid unintended harm.

Act early

Consider a new building being constructed downtown. A contractor has been called upon to build something that will stand for the next 75 years. To save costs, the contractor uses poor-quality cement. The owner doesn't invest in supervision since they want to avoid overhead to be able to move fast. The contractor continues building on top of that poor foundation and finishes the building on time.

Within a year, cracks start showing up and it appears that the building might topple. The city decides that this building poses a safety risk and requests for it to be demolished. The contractor's decision to save cost and the owner's decision to save time in the beginning now end up costing the owner much more money and time.

You might encounter this narrative often in ML systems. Companies might decide to bypass ethical issues in ML models to save cost and time, only to discover risks in the future when they end up costing a lot more, such as the preceding case studies of Ofqual and Strava.

The earlier in the development cycle of an ML system that you can start thinking about how this system will affect the life of users and what biases your system might have, the cheaper it will be to address these biases. A study by NASA shows that for software development, the cost of errors goes up by an order of magnitude at every stage of your project lifecycle.[29]

Create model cards

Model cards are short documents accompanying trained ML models that provide information on how these models were trained and evaluated. Model cards also disclose the context in which models are intended to be used, as well as their limitations.[30] According to the authors of the model card paper, "The goal of model cards is to standardize ethical practice and reporting by allowing stakeholders to compare candidate models for deployment across not only traditional evaluation metrics but also along the axes of ethical, inclusive, and fair considerations."

The following list has been adapted from content in the paper "Model Cards for Model Reporting" to show the information you might want to

report for your models:[31]

- *Model details*: Basic information about the model.
  - Person or organization developing model
  - Model date
  - Model version
  - Model type
  - Information about training algorithms, parameters, fairness constraints or other applied approaches, and features
  - Paper or other resource for more information
  - Citation details
  - License
  - Where to send questions or comments about the model
- *Intended use*: Use cases that were envisioned during development.
  - Primary intended uses
  - Primary intended users
  - Out-of-scope use cases
- *Factors*: Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others.
  - Relevant factors
  - Evaluation factors
- *Metrics*: Metrics should be chosen to reflect potential real-world impacts of the model.
  - Model performance measures
  - Decision thresholds
  - Variation approaches
- *Evaluation data*: Details on the dataset(s) used for the quantitative analyses in the card.
  - Datasets
  - Motivation
  - Preprocessing
- *Training data*: May not be possible to provide in practice. When possible, this section should mirror Evaluation Data. If such detail is not possible, minimal allowable information should be provided here, such as details of the distribution over various factors in the training datasets.
- *Quantitative analyses*
  - Unitary results
  - Intersectional results

- *Ethical considerations*
- *Caveats and recommendations*

Model cards are a step toward increasing transparency into the development of ML models. They are especially important in cases where people who use a model aren't the same people who developed this model.

Note that model cards will need to be updated whenever a model is updated. For models that update frequently, this can create quite an overhead for data scientists if model cards are created manually. Therefore, it's important to have tools to automatically generate model cards, either by leveraging the model card generation feature of tools like TensorFlow, Metaflow, and scikit-learn or by building this feature in-house. Because the information that should be tracked in a model's card overlaps with the information that should be tracked by a model store, I wouldn't be surprised if in the near future, model stores evolve to automatically generate model cards.

Establish processes for mitigating biases

Building responsible AI is a complex process, and the more ad hoc the process is, the more room there is for errors. It's important for businesses to establish systematic processes for making their ML systems responsible.

You might want to create a portfolio of internal tools easily accessible by different stakeholders. Big corporations have tool sets that you can reference. For example, Google has published recommended best practices for responsible AI and IBM has open-sourced AI Fairness 360, which contains a set of metrics, explanations, and algorithms to mitigate bias in datasets and models. You might also consider using third-party audits.

Stay up-to-date on responsible AI

AI is a fast-moving field. New sources of biases in AI are constantly being discovered, and new challenges for responsible AI constantly emerge. Novel techniques to combat these biases and challenges are actively being developed. It's important to stay up-to-date with the latest research in responsible AI. You might want to follow the ACM FAccT Conference, the

Partnership on AI, the Alan Turing Institute's Fairness, Transparency, Privacy group, and the AI Now Institute.

Summary

Despite the technical nature of ML solutions, designing ML systems can't be confined in the technical domain. They are developed by humans, used by humans, and leave their marks in society. In this chapter, we deviated from the technical theme of the last eight chapters to focus on the human side of ML.

We first focused on how the probabilistic, mostly correct, and high-latency nature of ML systems can affect user experience in various ways. The probabilistic nature can lead to inconsistency in user experience, which can cause frustration—"Hey, I just saw this option right here, and now I can't find it anywhere." The mostly correct nature of an ML system might render it useless if users can't easily fix these predictions to be correct. To counter this, you might want to show users multiple "most correct" predictions for the same input, in the hope that at least one of them will be correct.

Building an ML system often requires multiple skill sets, and an organization might wonder how to distribute these required skill sets: to involve different teams with different skill sets or to expect the same team (e.g., data scientists) to have all the skills. We explored the pros and cons of both approaches. The main cons of the first approach is overhead in communication. The main cons of the second approach is that it's difficult to hire data scientists who can own the process of developing an ML system end-to-end. Even if they can, they might not be happy doing it. However, the second approach might be possible if these end-to-end data scientists are provided with sufficient tools and infrastructure, which was the focus of Chapter 10.

We ended the chapter with what I believe to be the most important topic of this book: responsible AI. Responsible AI is no longer just an abstraction, but an essential practice in today's ML industry that merits urgent actions. Incorporating ethics principles into your modeling and organizational practices will not only help you distinguish yourself as a professional and cutting-edge data scientist and ML engineer but also help your

organization gain trust from your customers and users. It will also help your organization obtain a competitive edge in the market as more and more customers and users emphasize their need for responsible AI products and services.

It is important to not treat this responsible AI as merely a checkbox ticking activity that we undertake to meet compliance requirements for our organization. It's true that the framework proposed in this chapter will help you meet the compliance requirements for your organization, but it won't be a replacement for critical thinking on whether a product or service should be built in the first place.

1 Sometimes, you can get different results if you run the same model on the same input twice *at the exact same time*.

2 Eugene Yan, "Unpopular Opinion—Data Scientists Should be More End-to-End," EugeneYan.com, August 9, 2020, *https://oreil.ly/A6oPi*.

3 Eric Colson, "Beware the Data Science Pin Factory: The Power of the Full-Stack Data Science Generalist and the Perils of Division of Labor Through Function," MultiThreaded, March 11, 2019, *https://oreil.ly/m6WWu*.

4 Erik Bernhardsson on Twitter (@bernhardsson), July 20, 2021, *https://oreil.ly/7X4J9*.

5 Colson, "Beware the Data Science Pin Factory."

6 "Full Cycle Developers at Netflix—Operate What You Build," *Netflix Technology Blog*, May 17, 2018, *https://oreil.ly/iYgQs*.

7 Elliot Jones and Cansu Safak, "Can Algorithms Ever Make the Grade?" *Ada Lovelace Institute Blog*, 2020, *https://oreil.ly/ztTxR*.

8 Tom Simonite, "Skewed Grading Algorithms Fuel Backlash Beyond the Classroom," *Wired*, August 19, 2020, *https://oreil.ly/GFRet*.

9 Ofqual, "Awarding GCSE, AS & A Levels in Summer 2020: Interim Report," Gov.uk, August 13, 2020, *https://oreil.ly/r22iz*.

10 Ofqual, "Awarding GCSE, AS & A levels."

11 Jones and Safak, "Can Algorithms Ever Make the Grade?"

12 Jones and Safak, “Can Algorithms Ever Make the Grade?”

13 Ofqual, “Awarding GCSE, AS & A Levels.”

14 Jones and Safak, “Can Algorithms Ever Make the Grade?”

15 “Royal Statistical Society Response to the House of Commons Education Select Committee Call for Evidence: The Impact of COVID-19 on Education and Children’s Services Inquiry,” Royal Statistical Society, June 8, 2020, *https://oreil.ly/ernho*.

16 “Guidance on the Protection of Personal Identifiable Information,” US Department of Labor, *https://oreil.ly/FokAV*.

17 Sasha Lekach, “Strava’s Fitness Heatmap Has a Major Security Problem for the Military,” *Mashable*, January 28, 2018, *https://oreil.ly/9ogYx*.

18 Jeremy Hsu, “The Strava Heat Map and the End of Secrets,” *Wired*, January 29, 2018, *https://oreil.ly/mB0GD*.

19 Matt Burgess, “Strava’s Heatmap Data Lets Anyone See the Names of People Exercising on Military Bases,” *Wired*, January 30, 2018, *https://oreil.ly/eJPdj*.

20 Matt Burgess, “Strava’s Heatmap Data Lets Anyone See”; Rosie Spinks, “Using a Fitness App Taught Me the Scary Truth About Why Privacy Settings Are a Feminist Issue,” *Quartz*, August 1, 2017, *https://oreil.ly/DO3WR*.

21 “Fitness App Strava Lights Up Staff at Military Bases,” *BBC News*, January 29, 2018, *https://oreil.ly/hXwpN*.

22 Matt Burgess, “Strava’s Heatmap Data Lets Anyone See.”

23 Michael Feldman, Sorelle Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian, “Certifying and Removing Disparate Impact,” *arXiv*, July 16, 2015, *https://oreil.ly/FjSve*.

24 Wikipedia, s.v. “Differential privacy,” *https://oreil.ly/UcxzZ*.

25 Eugene Bagdasaryan and Vitaly Shmatikov, “Differential Privacy Has Disparate Impact on Model Accuracy,” *arXiv*, May 28, 2019, *https://oreil.ly/nrJGK*.

26 Sarah Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome, “What Do Compressed Deep Neural Networks Forget?” *arXiv*, November 13, 2019, *https://oreil.ly/bgfFX*.

27 Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton, "Characterising Bias in Compressed Models," *arXiv*, October 6, 2020, *https://oreil.ly/ZTI72*.

28 Hooker et al., "Characterising Bias in Compressed Models."

29 Jonette M. Stecklein, Jim Dabney, Brandon Dick, Bill Haskins, Randy Lovell, and Gregory Moroney, "Error Cost Escalation Through the Project Life Cycle," NASA Technical Reports Server (NTRS), *https://oreil.ly/edzaB*.

30 Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru, "Model Cards for Model Reporting," *arXiv*, October 5, 2018, *https://oreil.ly/COpah*.

31 Mitchell et al., "Model Cards for Model Reporting."