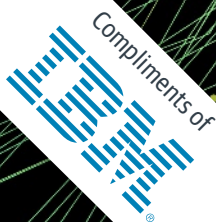


O'REILLY®

Compliments of  


# Agile AI

A Practical Guide to Building AI  
Applications and Teams

Carlo Appugliese, Paco Nathan  
& William S. Roberts

REPORT

---

# Agile AI

*A Practical Guide to Building AI  
Applications and Teams*

*Carlo Appugliese, Paco Nathan,  
and William S. Roberts*

Beijing • Boston • Farnham • Sebastopol • Tokyo

**O'REILLY®**

## Agile AI

by Carlo Appugliese, Paco Nathan, and William S. Roberts

Copyright © 2020 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://oreilly.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or [corporate@oreilly.com](mailto:corporate@oreilly.com).

**Acquisitions Editor:** Rachel Roumeliotis

**Development Editor:** Sarah Grey

**Production Editor:** Beth Kelly

**Copyeditor:** Octal Publishing, LLC.

**Proofreader:** Sharon Wilkey

**Interior Designer:** David Futato

**Cover Designer:** Karen Montgomery

**Illustrator:** Rebecca Demarest

October 2019: First Edition

### Revision History for the First Edition

2019-10-10: First Release

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Agile AI*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the authors, and do not represent the publisher's views. While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

This work is part of a collaboration between O'Reilly and IBM. See our [statement of editorial independence](#).

978-1-492-07495-3

[LSI]

---

# Table of Contents

<b>1. Introduction: Agile AI Processes and Outcomes.....</b>	<b>1</b>
The Agile Approach	3
AI Processes in Businesses Today	4
Beyond R&D	6
Organizing for AI	7
Why Agile for AI?	9
Summary	11
<b>2. Understanding AI Tools.....</b>	<b>13</b>
Contrasting Machine Learning and AI	13
The Role of Open Source in Innovation	15
Tooling	17
The Fundamentals of Machine Learning Projects	19
The Machine Learning Life Cycle	21
Distributed Workloads and Hybrid Environments	24
Summary	26
<b>3. AI Skills.....</b>	<b>27</b>
Understanding the Skills and Culture	28
Team Skills Assessment	29
Core Skills	31
Building Teams	37
<b>4. Summary.....</b>	<b>41</b>
Use Cases	41
Data	42
Tools and Process	43

Mindset	43
Integration and Trust	44
Conclusion	45

# Introduction: Agile AI Processes and Outcomes

With a claim in a potentially \$13 trillion market<sup>1</sup> at stake over the next decade, companies are working diligently to take advantage of the high returns of embedding artificial intelligence (AI) in their business processes—but project cost and failure rates are on the rise. Problematically, there is no standard practice for how to implement AI in your business. That makes it very difficult for business leaders to reduce their risk of project failure.

Although this is a book about potential *project* failures, we'd be remiss not to point out the potential damage of *organizational* failures. An organization fails without a corporate will to even consider AI projects because its leadership doesn't understand how far behind they are from industry leaders. To use a bike race as a simplified metaphor: in AI, industry leaders are the small number of riders in the breakaway, most businesses are in the peloton, and those without the will to adopt AI forgot they were in the race and are watching on TV. We don't speak to building institutionalized beliefs that enable you to avoid organizational failures in this book, but we do help AI projects address failure.

---

<sup>1</sup> McKinsey & Company, "AI Adoption Advances, but Foundational Barriers Remain," November 2018; T. Fountaine, B. McCarthy, T. Saleh, "Building the AI Powered Organization", *Harvard Business Review* (2019); S. Ransbotham, D. Kiron, P. Gerbert, M. Reeves, "Reshaping Business with Artificial Intelligence", *MIT Sloan Management Review* (2017).

Why are project failure rates so high? There are three areas in which things can go wrong (see [Figure 1-1](#)):

### *Skills*

First, high skill levels are needed to harness AI.

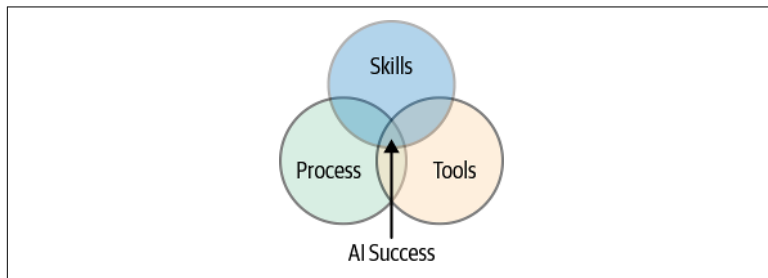
### *Process*

Every use case can be developed in a different way. There is no blueprint for developing AI applications and integrating them into your current business workflow.

### *Tools*

There are hundreds of open source and proprietary AI tools. Each can have hidden limitations and costs. There is no guarantee of success.

Skills, processes, and tools are like a three-legged stool: if just one of the three legs fails, the stool falls over. With this in mind, this book offers an agile AI approach for business that will allow you to innovate quickly and reduce your risk of failure.



*Figure 1-1. The three pillars of AI success*

This book was written with technical leaders in mind, especially those who have an understanding of data warehousing, analytics, software engineering, and data science. But anyone interested in an Agile approach to AI for business will find it useful.

In 2017, IBM created the IBM Data Science Elite (DSE) team, a team of the world's top data scientists, to help clients on their AI journey. We have used AI in hundreds of projects to bring real business value to our clients quickly. We've built a unique Agile methodology using AI and teamed up with leaders in the open source space. We've written this book as a guide for business leaders and all technical teams working on AI projects.

In hundreds of conversations about data science that we've conducted with practitioners, we've come to learn that there is no singular way to pursue AI to achieve their analytics goals. There are only quicker ways or longer ways. Some businesses are just picking up data science, and some are deep into their analytics journeys, but their approaches all differ. In this book, we talk about those journeys and aim to impress upon you the value of an *Agile AI* practice.

## The Agile Approach

*Agile* is a software development methodology designed to address the shortfalls of antiquated Waterfall development systems. In old software development practices, a team would scope a product a length of time into the future and build work-back plans. Each separate participatory team would address those capability requirements week by week and then hand off their work to the next group.

For example, to build my factory widget-counting software product, I know I need components A through C. The design team comes up with the design of A, hands that to the product team, and then the product team scopes my work for the engineering team. The design team then picks up component B, and so on.

In new software development practices (*Agile*), teams work cross-functionally to design minimally viable products (MVPs) and ship iterations of those products quickly. The handoff becomes less of a waterfall, in which one team's work flows downstream—cascading, if you will—to the next group. With *Agile*, the handoff is much more collaborative.

Let's now look at building my factory widget-counting software product from an *Agile* perspective. I know that I need components A through C. Design works with the product ownership team and engineering to build and certify the MVP of component A, while another cross-functional team can pick up component B.

*Agile* methodology tends to work so well because the process of building software inevitably runs into unexpected roadblocks (i.e., one group doesn't have sufficient permissions, infrastructure, or budget). Delays happen. Bugs pop up. By building MVPs in smaller chunks, you're not going to push off the entire development timeline as you would when one group botches its delivery date in handing off to a separate siloed group.



IBM has incorporated a unique Agile software development life cycle (SDLC) approach in our engagements with clients, but focused on building AI products. In it we use sprint development principles, work on cross-functional teams, and focus on readouts. *Sprints* are two-week-long time boxes, and *readouts* are times to “show & tell” your work. Within each sprint, we prioritize understanding the business problem, engineering data features based on our in-depth understanding of the business problem, and then we quickly move to train and test our models based on that understanding. These iterative cycles allow us space for creativity as well establish a rhythm for fast-paced problem solving. If we’re able to explain as much of the variance in our data as we need, we’re done! Otherwise, when we run into roadblocks with finding signal among the noise in our data, we can shift our focus, include new insights from the business, and then begin to perform feature engineering all over again.

**NOTE**

There are some basic statistical principles you need to account for as you go through these iterations so that you don’t invalidate your hypotheses.

This process—with its incorporated principles of design and solid fundamentals of software development—has worked time after time and is something we love to talk to leaders about.

## AI Processes in Businesses Today

When we hear clients or coworkers discussing AI, our first thought is to ask them to define what AI means to them so that we can have a shared understanding. Lots of people think of AI as magic. Think of it instead as *analytics that you and your enterprise depend upon to make decisions to drive your business*. The term *artificial intelligence* reflects a broad practice of systems that mimic human intelligence to make informed business decisions based on your company data. At the heart of human intelligence is the ability to learn, which means that AI must involve mastering and democratizing machine learning. If you and your team have used predictive analytics to optimize your business workflows, you might be as close to AI as anyone else out there.

AI has not yet reached the capacities of generalized intelligence; that is, self-empowered, decision-making computers: an AI system with the ability to reason and decide with the same capacity of an industry expert. Currently, it produces decisions informed by the historical data stored over the years by your enterprise. Generalized AI will be achieved at some point as tools and business applications develop, but it might be that every industry will reach that milestone in different time frames. For example, AI researchers **hypothesize** that AI will automate all retail jobs in 20 years, whereas automating human surgeries will take 40 years. But using AI for your business? That is something you can do today—and something you *need* to do today.

When we began working with the credit-reporting service Experian, we discussed some of its challenges with entity matching to build a corporate hierarchy for all companies. Experian's existing process required a lot of human capital and time, and that limited its ability to match entities across its full population. Although the company understood AI, the best approach for this specific line of business was not clear, so we worked with developers on small steps and in quick iterations. The companies leading the way in adopting AI understand the value of incremental changes, and Experian became one of these successes.



*Entity matching* is resolving records of a database and deciding which align to one another. In the corporate hierarchy application, the question is framed as “which two businesses (and, importantly, their subsidiaries) are the same?”

When we share our experience with clients, we push them to move fast: if you're going to fail, you should do it quickly. You don't really know whether your approach will work until you model it with data. So, you need to invest. It makes client projects more successful because we test the theories we suppose to be true, and if those theories aren't borne out in data, we quickly discover that and adjust.

Contrast this development cycle to a traditional modeling life cycle—think of the Waterfall design of software development in the past, with the antiquated premise that good software can go through a “design first, deliver final” process instead of iterating on work constantly.

Each of these approaches (Agile and traditional) was a decision our clients made intentionally, and each is valid. Ultimately, the difference distills down to the rate of adoption of each enterprise's AI use case.

## Beyond R&D

The most effective campaigns to achieve AI at enterprise scale tend to focus related projects on business objectives. Let's compare the different ways companies are approaching AI to help us understand data science compared to R&D.

If machine learning is the language behind AI, statistics are the grammar of that language. Enterprises have long had analysts reviewing historical data in search of trends using statistical principles that are useful for reports and experimentation. Those reports and experiments are one function of advising the business, and business analysts have achieved great success utilizing them, but often the report is the final—or only—outcome of that wider analysis. Predictive capabilities for AI, on the other hand, must be wrapped into broader application deployments and made available.

By comparison, when they participate in conversations about AI, many enterprises think only of their research and development function. For line-of-business leaders, AI can feel unachievable, so they might believe that whatever comes out of the research organization will be the closest thing to AI that their enterprise will achieve. Predictive algorithms are improving all the time, thanks to R&D, and many organizations, from IBM to McKinsey, to Google, have excellent research functions to do just that. AI, however, is not the improvement of a function; rather, it's the *broad and strategic application of a set of predictive capabilities designed up front to solve difficult business problems*.

As we look to drive business outcomes, IBM data science teams embed with client teams working to solve the business problems at hand, not their R&D functions. As we build out their AI capabilities, we design our applications to operationalize the models from the very beginning. It is common for enterprises, and our clients, to struggle to make substantive use of their predictive solutions because they don't begin their data science exercise with the intention of consuming the models they've spent many hours training. If enterprises are to achieve AI or the ability to apply predictive

features in every function where it is economically feasible, they need to begin each problem-solving exercise with the expectation that their modeling will be consumed by some other application elsewhere.

Leaders like Quicken Loans that adopt an innovation culture confirm our view that innovation occurs when it is focused on business problems and embedded in each line of business. We've seen many enterprise companies treat AI as only a scholastic science fair project or as a research and development project. Research is essential and can make our applications better, more accurate, or more efficient, but it is the widespread and ease-of-use tooling of predictive capability that has enabled their AI. R&D might have improved it, and analysis might have shown that it was effective, but designing their predictive capabilities with business problems in mind is what makes those solutions feasible.

## Organizing for AI

We frequently hear from technical leaders when they struggle with how to best organize the talent within their teams to enable AI development to thrive. This is always a revelatory conversation.

First of all, when the IBM DSE team begins any engagement, we mandate that each client bring data scientists from within their organization to the table. We strive to help grow skilled data science practitioners; if we didn't, each engagement would fail as soon as IBM left. Those data scientists can then serve as *Centers of Excellence* for other groups or lines of business needing to build AI solutions. It was never intentionally decided that the DSE team would help lead enterprises in this direction, but ultimately, we help clients grow their data science practice. Clients ask all the time about how best to scale data science in their enterprise. Our answer is to build the appropriate organizational structure. Companies *can* innovate with separate pockets of data scientists serving different purposes in a decentralized model, but the *best* model is a hub-and-spoke model. The hub serves as a Center of Excellence focused on supporting all parts of your company, whereas the spokes are located in each line of business focused on business problems.

## Data Scientists

For each use case you will implement, each project team needs a variety of skills. First, the data scientists tend to be generalists with a toolbox, which allows them to serve as service centers. The principles of their practice endow them with comfort in handling many different sets and types of data and applying best practices when it comes to cleaning, preparing, and understanding that data. They are mandated to maintain an understanding of the state of the art in the field, but as it applies broadly to data science.

## Data Engineers

Data engineers are masters of the data pipeline, and they're essential for owning the data cleaning and architecture that data scientists depend upon. They also don't pray to any singular type of data, so their skill sets generalize well to teams in a broad sense.

## Business Analysts

By contrast, business analysts are often tied to business units and are experts at the intricacies and practices within the field of concern of that unit. They know data within their field well, and when a data scientist doesn't know what a specific feature means, they turn to the business analyst. When a data scientist doesn't understand a problem in any specific sense, it would likely be the business analyst to whom they turn.

There is a naturally cohesive partnership between the data engineers and data scientists, with transferable skills across many teams, whereas business analysts tend to remain embedded and act as a stationary Center of Excellence. Teams that are set up to build applications with full modeling and data science pipelines might think each business unit requires its own set of data engineers and scientists, but in the enterprise that tends to be overkill, especially when all enterprises are competing for relatively scarce data scientists and data engineers.

In summary, there are a few essential conditions for the hub-and-spoke model to work. The business analysts should know the business problem as well as the data, in detail. The data scientists should have the know-how to dissect that problem and data, and model an approach. The data engineers can then help with the data pipeline,

create the architecture of a solution, and model operations. Together, these clusters of talent and institutional domain expertise can work together to deliver AI solutions.

## Why Agile for AI?

Failure is good. And the best way to fail is to do it quickly.

Leaders, developers, and data scientists alike blindly believe in the dogma that adding predictive capabilities to their businesses is worth the effort only if it is an overnight success. That understanding is neither practical nor realistic when the reality is that an iterative approach is rooted in the scientific method of forming hypotheses and validating your theories. Not all theories or predictions can be proven to be true.

First, make sure you're working on the correct business problems. It is difficult to identify business cases for which AI solutions will drive impactful change to the business. Some business problems are not economical to solve with a model because gathering, cleaning, and storing the data are cost prohibitive. For example, we embarked on a project to use machine learning to reduce noise in file processing. That's a very common use case, and machine learning is very effective on these kinds of projects. Although the client was interested in using machine learning, when we quantified the business impact, the solution would save two resources 10 hours of manual work per week. The cost to do the project didn't justify the savings. We always tell clients that even though AI can transform your business, it's best to first use it on *big* problems until you have proper teams in place and can democratize the technology across your company.

Second, for some use cases, the existing data doesn't carry the essential signal for modeling efforts. Data scientists can save all the data they want, but if there is no discernible historic relationship between the data and the outcome, it will be impossible to predict the behavior of interest. This isn't too common: typically, we can perform additional feature engineering and pull in more relevant data to get signal in our models. But it does happen, so you need to understand it as a risk and adjust your use case and approach when faced with it.

The salient point for all enterprises interested in modeling from their data is that it is important to try to model many behaviors, quickly. Leaders should establish a broad swath of hypotheses for their teams and enable them to attempt each modeling experiment. Many times, when the IBM DSE team is brought in to solve a modeling problem, we see enterprises despair at the slightest roadblock. We always tell them: the first model snafu is not the end of the exercise. As long as we're responsible with how we treat the significance and power of our experiments, we can persevere.

As an example of perseverance, let's talk about a project with one of the largest automotive parts manufacturers in Canada: Spectra Premium. We were asked to help the company build a better forecasting model than a third-party tool it had been using. Spectra Premium manufactures many parts, but runs are relatively low volume. Each part takes a long time to manufacture because it requires special machinery adaptations during production. Because of the lengthy production time, forecasting accurately was critically important. If Spectra doesn't forecast the demand properly, it could potentially have items back-ordered and open the door for competitors. In our first few modeling attempts, we were only able to match the results that the company was already achieving, which was 80% accuracy. As a collective group, we were disappointed, but we dug deeper into the data and found that our accuracy for some manufactured parts was as low as 30%, whereas for other parts it was greater than 90%. We changed our approach and instead began to try different algorithms for each part instead of modeling on the entirety of the inventory. This approach pushed our *overall* accuracy to more than 90% and led to a successful example of the team adapting in the face of adversity.

In contrast to software engineering teams, data science teams can offer no guarantees about modeling outcomes. When your enterprise adds a traditional software application, the development life cycle is well-known and well-documented. Even then, leadership needs patience as their teams iterate over the product features and behaviors, but the exercise is not predicated upon an unforeseen set of data: its success is largely driven by time and scope. Data science exercises, however, all begin with a hypothesis. As in any experiment, sometimes your teams cannot reject the null hypothesis.

The faster your teams iterate through their experiments and test their hypotheses, the more likely you are to drive the amazing

outcomes that AI can offer enterprises. Your willingness and flexibility to pivot is a key factor in the success of your AI projects.

## Summary

The IBM DSE team built a unique Agile process for our clients' projects, and in this book, we draw from those experiences to show you how your teams can succeed with an Agile approach to AI.

Legendary science-fiction author Arthur C. Clarke famously wrote that “any sufficiently advanced technology is indistinguishable from magic.” Despite what you might have read, AI is not magic. You really need to understand this technology in order for your teams to harness it. This book is a chance for us to share the stories and key practices of our successful client engagements to help you succeed in your Agile AI practices.





# Understanding AI Tools

This chapter is about how we can understand the technology behind AI. We spend much of our time with clients talking about the differences between terms that are often used interchangeably. That’s an indication that it is valuable to differentiate here what they mean.

We also take the time to talk about how open source projects are critical to the tools and fundamental principles behind AI, and then talk about the trends and risks of machine learning. Both practitioners and leaders in the AI space need to be aware of the effects of models.

## Contrasting Machine Learning and AI

Before we discuss machine learning and AI, it’s important to clarify what each means and how to talk about them. It’s essential to do because we’re currently at the peak of the hype cycle for AI, and as a consequence a lot of companies are selling snake oil branded as “AI.” When a term is as overused as AI, people will continue to try to define it so that they can abuse it. So, first, let’s define what these terms and practices mean for technical leaders.

*Machine learning* is an engineering discipline: it’s the tooling and techniques involved here. At its core, machine learning is categorized as a form of mathematical optimization—because the algorithms are used to perform optimizations to create “learners” from training data. Keep in mind that algorithms are much less valuable than data. In the article [“Datasets Over Algorithms”](#), Alexander

Wissner-Gross showed that the mean time between a new machine learning algorithm being published and its use in an AI breakthrough was 18 years; however, the mean time between the required *datasets* becoming available and those AI breakthroughs was 3 years. Machine learning without the necessary data and use cases is merely a pile of nuts and bolts waiting to be built into something useful. Nonetheless, machine learning is about *learning* from data, not about writing code, and that represents a fundamental difference from previous software engineering practices.

*Artificial intelligence* is where the uses of machine learning begin to affect social systems. That can be where machine learning models perform tasks with human-like levels of proficiency, such as converting speech to text or recognizing tumors from radiology images. Even more likely, it can be where machine learning models augment teams of people, such as flagging potential credit card fraud that customer service agents need to explore further. Or it can be where machine learning is used for wildly nonhuman tasks such as making a computer science student appear on video to be able to dance as well as a professional ballerina—see “**Everybody Dance Now**” by Caroline Chan et al.—also known as *deep fakes*. In each case, the AI applications have effects on groups of people, and that’s how we measure their impact.

Note that we also talk about data science here. In general, we see *data science teams*, which tend to use *tools for machine learning*, and might develop *artificial intelligence applications*. When we’re talking about expert teams, we describe those as data science teams.

To use a *Star Wars* analogy, R2-D2 is a robot that takes in data, learns, and makes decisions, but he was built to interface with an X-wing star fighter, aiding as a copilot. Think of R2-D2 as machine learning. C-3P0 is also a robot that takes in data, learns, and make decisions, but he can speak and is programmed to understand millions of languages. He aids humans, but in more general ways than R2-D2. C-3P0 can replicate human intelligence but in many cases exceeds human performance. Think of C-3P0 as artificial intelligence. What about deep learning? Think of Yoda, who operates at a deeper level using the force (neural networks). This is mostly a joke, but it’s an easy way to describe the relationship between machine learning and AI.

How do these technologies fit in with software development? In some cases, software applications are sufficient for solving a problem directly. For example, if I get a sales lead onto a queue from marketing, I'll create an application to automatically append information like the lead's zip code and account information and then give it to the sales representative. By comparison, *machine learning* refers to applications that depend on probabilistic algorithms applied in a programming paradigm. With machine learning, if I get a sales lead onto a queue from marketing, I'll send the lead to my machine learning model, which was trained on historical sales data with an algorithm (such as the **random forest algorithm**) to help me decide how likely the lead is to purchase my product. In other words, "How hot is this lead?" These are questions without definite answers; however, we can offer predictions with some range of certainty.

As you might imagine, there are myriad business problems that have no definite answers. Intuitively, when a person is asked to make an estimate, we often think of answers in ranges or intervals. When you train machine learning models to provide answers, you begin to build out AI for your business.

## The Role of Open Source in Innovation

In the data science space, open source is king. Groups like NumFOCUS have created powerful and sustainable libraries and tools that have changed the scientific computing and data science landscape for the better. Projects like NumPy, SciPy, pandas, and Project Jupyter are integral to every data scientist's workflow and are the standard to which all other data science projects are now compared.

As a testimony to the success of these projects, at IBM we've modeled our data science products on their revelations. In Watson Studio, for example, we've incorporated an interface of stacked executable cells like the notebooks from Project Jupyter, and we actively support the pandas, NumPy, and SciPy libraries. In other companies, it's the same thing: Microsoft Notebooks and Databricks Workspace, to name a couple. At a time when the scientific computing space was fractured and without leadership, NumFOCUS stepped in to standardize the libraries and APIs the vast majority of data science now depends on.

Open source as an ideology has changed the way companies around the world build products. The premise is that people give their time freely to build tools and libraries that solve problems that are important to them and their work. Red Hat (now part of IBM) decided to use that in its business model by taking a fork of the code of those open source projects, adding features important to its enterprise customer base, and charging for support and consulting services. Companies can take forks of these projects and build them directly into features of their products.

The software community as a whole depends increasingly on open source projects, and if your company uses software, it probably does too. To innovate in your business, you need to raise your open source social score. Community-sourced projects have maintainers and core contributors who direct their projects, but they aren't beholden to any singular corporate entity. The projects bring into their development pipeline the features that are important to them and only them. As a consequence, the system can appear to outside observers to be riddled with chaos. To enjoy the benefits of the capable and qualified engineers solving these complicated problems, leaders and companies need to embrace the chaos of community-driven development.

It's important to clarify that open source doesn't necessarily mean free. Open source project usage and maintenance still requires upkeep and maintenance in your own applications. Those projects are not always suited for regulated production environments. If Red Hat is an example, consider that its value-adds are in major part based on hardening the security of these community-sourced projects. They turn those projects into offerings for the enterprise, and many companies have embraced Red Hat as a way to trust open source innovation in their enterprises. Additionally, open source projects mean that maintainers and members of the community are in charge of the project's direction instead of you. When and if a community chooses to pick up feature requests that are important to you, it might be slow in doing so. Those adoption costs are factors you need to consider.

# Tooling

There are many open source frameworks and tools for machine learning that are appropriate for enterprise use cases. [Figure 2-1](#) gives some context for the AI technologies that you'll need to build your models.

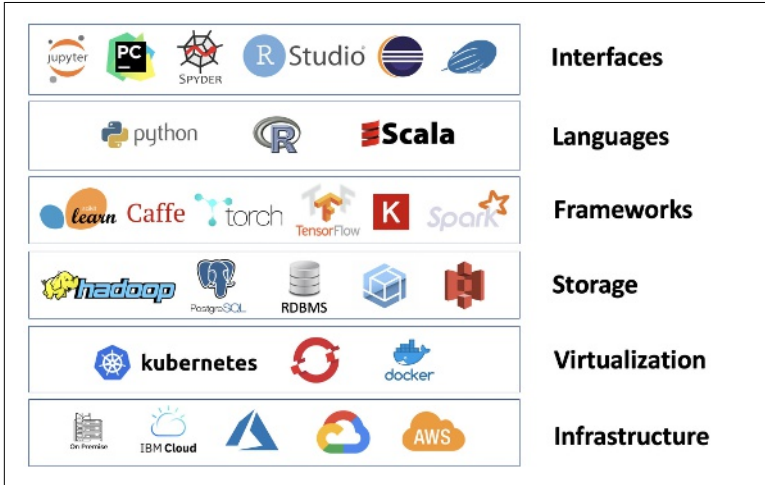


Figure 2-1. Technology stack

This is not an exhaustive list of tooling; it's a list of tools rather than specific data science packages. In other words, we've identified that Jupyter notebooks are important as a tool without qualifying that, to work efficiently, a data science team will need to depend on libraries and frameworks like TensorFlow or scikit-learn. Additionally, Python and R are commonly used languages, and each has a variety of usable workbenches. Besides Jupyter Notebook, graphical user interfaces (GUIs) are commonly used, such as PyCharm for Python and R-Studio for R. A notebook, however, is entirely sufficient for executing exploratory data analysis (EDA), model training, and model evaluation.

Data science in large part deals with cleaning and evaluating large stores of data. Data science as a discipline took off in popularity concurrent with advances in capabilities to store and process data—namely, Hadoop Distributed File System (HDFS) and map-reduce as the storage and processing mechanisms, respectively. As it became more economical to store and evaluate data, the practice of

utilizing it matured. In light of that, the open source solutions for terabyte-scale data persistence will likely revolve around HDFS and what many term Hadoop's successor: Spark. Spark differs from Hadoop in that it has figured out how to load data, at scale, into memory for processing. By comparison, Hadoop depends on data read from storage. The difference in the upper bounds of processing speed, in comparing data read storage as compared to data read from memory, are drastic. Spark's gains in computational speed have buoyed its recent popularity and made it an essential part of the open source data science stack.

Although Hadoop and Spark are used extensively in AI projects, traditional relational database management systems (RDBMSs) are still relevant today for data science. Many organizations keep their trusted data in these managed systems, and data scientists perform extensive structured query language (SQL) commands to extract the information they need.

After we've built a machine learning model, it must be deployed for the intended use case. Sometimes, that involves deploying a model as a *microservice* and scaling to meet customer demand. That might involve Kubernetes, autoscaling, load balancing, and other network engineering work. Other times, we must run a machine learning model directly on a smartphone and other mobile devices. For that, we need to use *model compression* before loading the model into the embedded hardware. In either case, there are tools for managing the model deployment and then monitoring model performance in production.

A new field is emerging, called *MLOps*, for the operations work required to manage the full end-to-end machine learning life cycle. In other words, the definition of *operations* has had to change to meet the needs of machine learning. Also note that the AI space is replete with vendors attempting to own differing segments or steps in the tooling. A lot of companies have perspectives on the full pipeline, and solutions to accommodate storage, model training, and model evaluation. An evaluation of vendors is beyond the scope of this ebook; here, we simply offer our perspective on the minimum required to staff and build in an AI pipeline.

# The Fundamentals of Machine Learning Projects

To build a data science team and lead it with authority, you need to know the fundamentals of a data science project and understand what they mean for your data science team's life cycle. Here, we talk about the different types of machine learning that underlie the foundation for AI as well as the life cycle of a data science project.

There are traditionally three types of machine learning that you'll need to be familiar with: supervised learning, unsupervised learning, and deep learning.

## Supervised Learning

*Supervised learning* is a subset of machine learning in which your training data has a target feature (also called a *label*) that you're trying to predict. As an example, if we were trying to create a model to predict the balance in a client's bank account, the dollar balance would be the target variable. Models match input values to output values based on historical information, and learn patterns within the dataset to most closely approximate the labeled value when making a prediction. There are many applications of supervised learning as well as many models within supervised learning to help achieve your modeling goals. Supervised learning models include *linear regression*, *logistic regression*, *decision trees*, *random forests*, and *boosted trees*, to name just a few. Whenever your data has a target value, you are going to use supervised learning to predict your target outcome.

## Unsupervised Learning

*Unsupervised learning* is the subset of machine learning in which your training data does *not* have a label for you to predict, and so your goal is to model likeness between the different training examples. Unsupervised models find patterns of similarity across the training samples, and attempt to cluster them together. In grouping data points together, we can learn a lot about their likeness and natural belonging. We can use unsupervised learning in applications that need capabilities like document/record classification, market segmentation, and data compression. Consider the example of classifying handwritten digits. Each training row would represent a vec-



tor of the pixels' intensity in the photo of the digit, and each cluster would reflect the model predicting an image to be one of the digits we're classifying. These unsupervised models aim to associate like data points because they lack the target/label to serve as the *ground truth* against which data scientists measure the success of the model. Importantly, without ground truth metrics, it is also implied that clusters do not need to be the same after each training run. Even with the same training data set, a model can offer novel groupings after each subsequent training run.

## Deep Learning

Figure 2-2 depicts how *deep* learning, based on *artificial neural networks*, is part of the broader scope of machine learning methods. Deep learning can be used as supervised learning or unsupervised learning—or even a newly emerging category of *self-supervised learning* that's beginning to be used to describe *generative adversarial networks* (GANs).

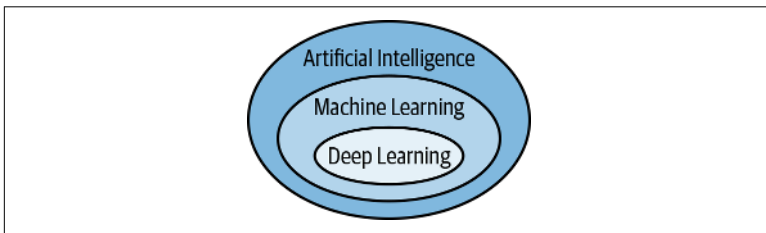


Figure 2-2. Deep learning's place within the AI sphere

The main point is that deep learning architectures have several layers of neural networks, which in practice causes certain properties and behaviors. For example, deep learning models can train without “saturating” such that millions of data examples can be used—unlike earlier forms of machine learning. This, in turn, allows deep learning models greatly increased accuracy, as observed in use cases for computer vision, speech recognition, natural language, and so on.

A good way to understand approximately how deep learning works is to consider the layers in an analogy for human vision. People have retinas on the back of their eyes. The retina has rods and cones that detect contrast and colors. At this lowest layer, human neural networks are taking in raw data. Then as the neurons connect further and further into the brain, processing progresses from raw input to higher-level concepts and actions: person, clown, danger, run! The

lowest layers of neural networks are vectors of raw data. Each layer organizes the data further, until the top layers can be resolved into the labels on the data; in other words, progressing from pixels in a radiology image to a decision about where a tumor is located. After a deep learning model has been trained, we can “lock” the lower layers and fine-tune the upper layers with new labels; for example, using a model originally trained on images of animals to fine-tune a new model that recognizes lung tumors. This process is called *transfer learning*, and it’s where we begin to see really interesting uses for AI.

## The Machine Learning Life Cycle

Let’s look at the life cycle of a machine learning project in order to better understand how to develop predictive solutions, regardless of machine learning model type. The life cycle consists of three distinct goals or processes: training, deployment, and running, as illustrated in [Figure 2-3](#).

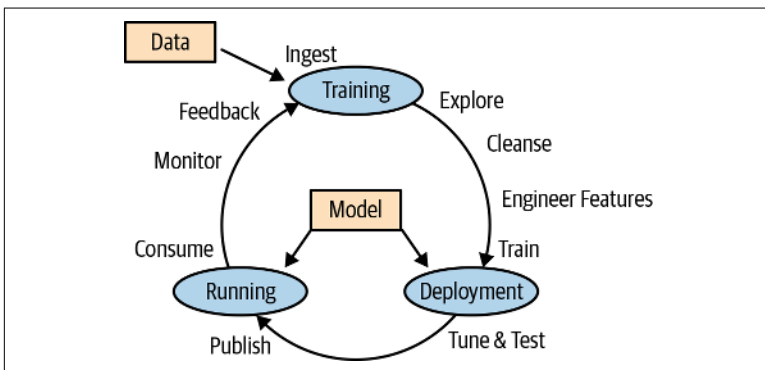


Figure 2-3. The machine learning life cycle

### Training

In the *training* part of the process, a data scientist takes their dataset and evaluates multiple models to identify which is the most effective for their predictive needs. Some models are easier to evaluate and understand, and offer greater parsimony: consider (regularized) linear regression. Some models are more effective at gathering predictive signal, and more complex to explain or implement: consider boosted trees.

For industries in which the business needs to be able to explain the choices its models have made, the power of parsimony makes simpler models appealing. Consider a model that evaluates whether to lend to homeowners to finance solar panel installations. If the model rejects applications for loans, your institution needs to be able to explain the reasons behind that decision. In applications that don't affect people, your requirements for explainability are less stringent. Consider a model that predicts whether your manufacturing line is going to experience a failure in equipment. If the model catches a sufficient number of failures in advance of them occurring, compared against a baseline metric of import to you, you won't feel obligated to understand the logic of each prediction made.

The training process is complicated not because of the model evaluation phase, but rather the steps preceding it: data cleansing and preparation. Data scientists spend the vast majority of their time working to prepare their data for modeling. They toil over understanding, cleaning, standardizing, normalizing, and feature engineering, and this makes sense. From a data science perspective, the greatest value-add in the process is in feature engineering.

*Feature engineering* is the part of the training process in which data scientists, with clean data, imbue domain expertise into their dataset by deriving new columns of data. Let's consider a fictional scenario of a data scientist modeling home prices. The data scientist can guess that specific home features are indicative of the value of a home—the more bathrooms a house has, the greater the price. To further capture the signal in their data, they consult with an expert in the field of selling homes. The realtor explains that it is actually the ratio of bathroom square footage to bedroom square footage that affects the price of a home. In their estimation, when a house has big bedrooms and tiny bathrooms, it is much less valuable. Now the data scientist needs to engineer that feature and add a column to their dataset: bathroom/bedroom ratio (sq ft). In that process, the data scientist has captured domain expertise in their solution by creating a derived piece of data. It is an expressive, creative, and powerful part of the training process.

## Deployment

The second part of the data science life cycle is *deployment*, wherein the trained model must be made consumable. Data science and building AI solutions is a powerful practice, but of little to no value

to your business if it can't consume the result of the training process. There are many ways to build an infrastructure around your models, but because we've discussed tooling, we'll instead talk about the goals and principles of the process.

Model deployment is a step in which your data scientists, data engineers, and DevOps teams work together to expose a machine learning model as an API. An API can take many forms, but more often than not, in the AI space the API is offered as a REST service. A web protocol like REST demands and defines a standardized mode of consumption. By conforming to a standard communication protocol, you make it easy for developers in your business (and possibly external developers) to consume your data product. Each AI model you train and deploy is a data product, and people aren't going to become customers if it's difficult to use. (Some models are accessed as batch deployments and therefore have a slightly different architecture. Although they're important to keep in mind, we don't address them here.)

The trained model is only a single component of the deployment an AI team is responsible for creating. Consider that a data scientist(s) provides the model after their data exploration, feature engineering, and training, but the model requires a data engineer to build a pipeline for the model. The data engineering team is responsible for curating or standardizing the input and output of data to the model, handling the routing of the data features that the model expects, and then returning the prediction the model produced. The data engineer, in essence, builds the more permanent persistence of the data going in and out of the predictive pipeline both for the sake of efficiency and retraining.

## Running

The final step in the data science life cycle is *running* your model to evaluate results, and that's where your application is made consumable. Traditionally in this step, a DevOps team takes the software application that the data engineers and data scientists have built and creates a standard for deploying it and making it consumable in applications and in production. Software applications require maintainability and scalability. Applications are only as successful as they are easy to deploy and maintain.

Running a software application used to be significantly more complex. Each application required routing handled by load balancers across many different application servers. That's still the case for many web-scale-sized app deployments, but the contemporary design nowadays for the kinds of solutions we've been discussing take advantage of containers and container orchestration frameworks.

A *container* is simply a standard unit (your model and REST application) that packages all of your configurations and dependencies into a single object, allowing flexibility and ease across runtime environments. With your predictive model and routing application bundled together, the DevOps team coordinates the environment for containers to grow and collapse as they are needed. In these contemporary designs, they don't fret over the constant uptime or maintenance of a single cluster of web servers and their respective single REST application. These architectures are more scalable and come with less overhead, which allows development teams to iterate with alacrity.

We've touched on what types of solutions data science teams create (training), how their counterparts in engineering (deployment) and DevOps handle their data products (running), and some of the reasons why. When it comes to runtime design—the responsibility of DevOps teams—take these suggestions with a grain of salt. There are *many* ways to handle runtime environments, and with the rate of innovation in the space, container orchestration could look like the last fad by the time you read this.

## Distributed Workloads and Hybrid Environments

Data science is an old practice, but the huge volume of data being used today is new, which is why data storage and compute are the basis for significant changes and exciting trends.

Historically, data storage was owned by several vendors who were able to serve consumers their data with great reliability and speed. Their profitability depended on how reliably they could return customer information; the more information customers wanted to store, the more it would cost them. To process more data, enterpri-

ses had to learn how to store it more cost-effectively. With the advent of HDFS, storage became cheaper.

Apache Hadoop was a project born of Yahoo's need to store web-scale data. The premise was that instead of using expensive hardware from large vendors, companies should be able to use cheap commodity hardware that is likely to fail. If you can account for the hardware failure by replicating the data, you can use cheaper hardware. The cheaper your hardware is, the more hardware you can use, and consequently the more data you can store. Data science teams started to store more and more data with the promise that they no longer needed to sample because they could now use as much data as they wanted to. *Data lakes* became popular as a result of this affordability in storage.

In response to a growing demand for storage and compute solutions to deal with the vast repositories in these data lakes, vendors have provided offerings to both house and analyze customer data. The emerging requirement in the past decade for distributed compute and storage put cloud offerings that simplified those pieces of the data analysis and modeling pipeline at the forefront of modern application architectures. As a logical derivative, the **trend** now appears to be headed toward building applications that rely on *serverless architecture*, in which there are no static virtualized environments, only systems responsible for computing resources.

**Serverless architectures** (Function-as-a-Service, or FaaS), as compared to the virtualized environments often thought of as Infrastructure-as-a-Service (IaaS), don't put the onus on developers to maintain an environment and yet allow them to take advantage of compute capabilities to which they otherwise wouldn't be privy. As an example, if our application were responsible for counting the widgets on our production floor at any one time, we'd send the actual application to a serverless service, the application would run, and we'd be charged by the **Platform-as-a-Service** (PaaS) provider for the amount of time our application takes to return us our final widget tally.

There are as many uses for serverless architecture as there are software applications, but there is also obvious appeal to data scientists and their teams who don't want to own the responsibility of managing or coordinating compute environments—the places they offload long-running jobs such as model training. It's in fact possible that

the FaaS offerings will serve as the de facto environments for all of our model training requirements moving forward. All we need to provide is access to the data, stored in either a private or public cloud, and the application that needs to execute. We would send our training script (with its access credentials to our historical widget production data) that's responsible for data cleansing procedures, model training, and cross-validation requirements, and then the serverless service would charge us for the amount of time our model takes to train and cross-validate. This is the basis for **papers** and **projects** stemming from the same **group** responsible for the Apache Spark project, which has revolutionized the way data scientists use large data in their applications.

This fits with broader industry trends in the cloud offerings world: we see companies working across public and private clouds and utilizing multiple clouds from varied cloud providers. From an ease-of-use perspective, there are few more promising and scalable technologies than these variations on cloud products. We're excited to see how they affect contemporary and future application designs.

## Summary

This chapter covered the fundamentals of data science and AI tools as well as provided an outline of possible trajectories to expect as these projects become more and more commoditized. You, as a part of a leadership team, have been armed with the knowledge of the terms and practices your teams will need as your enterprise aims to incorporate AI into its decision-making.

How can you assess the skills you need for a data science project? In some regard, no matter how agile your team is during your science project's duration, it won't matter if you don't have a minimal aptitude skill set within your team. In this chapter, we look at the core skills in data science and effective practices for building a data science team and nurturing a supportive culture.

There are a few roles that are almost always found on data science teams in industry. (Of course, some of the following descriptions might change, depending on the needs of a business vertical or specific focus.) We talk about these in general terms as follows:

### *Domain expertise*

Understanding the business needs and nuances within it; for example, regulatory compliance about data privacy if you work in health care.

### *Data science*

This is where the science and math come in—can you prove insights about the business based on advanced analysis of its data?

### *Coding*

Machine learning models need to be integrated into application software, and that requires programming.



## Systems

Data science tends to rely on lots of compute resources and requires people who are proficient with data engineering, distributed systems, and high-performance computing.

These typical roles indicate what kinds of skills are needed on a data science team. Before we dive into specifics, let's first take a look at some of the history that led to data science.

## Understanding the Skills and Culture

Data science and its culture trace back to 1962, with early definitions introduced in the paper “[The Future of Data Analysis](#)” by John Tukey. Prior to that point, statistics had been mostly a footnote within the larger scope of mathematics. It was initially about how nation states measured their economies, raised their armies, collected taxes, and planned new developments. With new capabilities for digital computing emerging in the 1960s, Tukey worked to make data analytics its own discipline, a scientific pursuit in and of itself. “We should seek out wholly new questions to be answered,” he wrote. People working in this new field had a responsibility to use computing resources to help make better judgments based on data plus applied math.

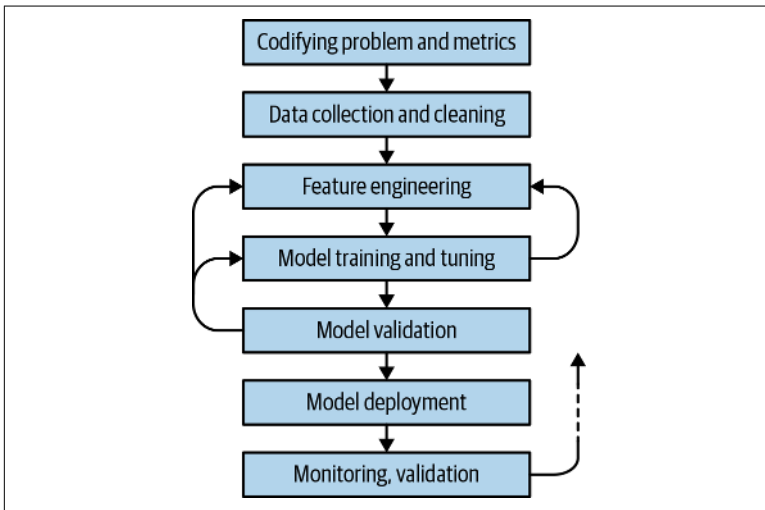
Already in 1962, we could see the outlines of data science team capabilities that are still current more than 50 years later. We use computing resources and manage data, which speaks to needs for data engineering. We use applied math, moving beyond academic debates and theoretical mathematics to address real-world problems. Algorithms and mathematical optimization play important roles, but they're only part of the puzzle. Domain expertise and firm grounding in the business use cases are also essential.

Tukey emphasized the word *judgment* in his introduction, addressing the fact that data analysis goes only so far toward solutions. Some problems can be fully automated, such as optimizing an advertising bid system, but the larger challenges in enterprise usually demand a careful blend of automation and human expertise. At the executive level, data science insights and machine learning models inform judgments, playing supporting roles while decision science moves to center stage. Even in that early sketch, data analysis was inherently interdisciplinary.

Today we take those definitions as givens. Successful data science teams are diverse by nature. Because no single person is likely to be an expert in several fields—computation, data management, applied math, business use cases, decision science, and so on—we build teams in which people with different skill sets and perspectives collaborate.

## Team Skills Assessment

A good way to map the skills needed on a data science team is to look carefully at the typical stages of a project. **Figure 3-1** describes a typical, idealized workflow for the different stages of a machine learning project.



*Figure 3-1. Typical stages in a machine learning project (Sophie Watson, William Benton, Red Hat)*

We examine each of these stages in terms of what skills are needed, but the first big questions to ask are what kinds of work will be needed in the project and which of these stages fit? For example, although most projects involve lots of data preparation, some might never need to move beyond reporting or visualization—enough work is invested so that other people in the organization can begin to see trends or other effects in the data. Other projects might require data science teams to lend their expertise in developing a narrative—for data storytelling—and other metrics that might then

feed into executive discussions and help inform organization decisions.

Other kinds of projects for a data science team involve systems for automated decisions, resulting in machine learning models to be used in products and services. What kind of services are needed: recommender systems? Anti-fraud models? Customer churn models? In other cases, human-in-the-loop approaches such as semi-supervised learning might be needed, in which machine learning models handle cases where they can predict patterns with some threshold of certainty but kick the problem back to human experts when predictions are too uncertain—blending the best of both people and automation.

For example, we discussed Experian in [Chapter 1](#). The company processes high volumes of files from banks about payment history for every person who has credit. Each day, hundreds of files fail because of a rules-based system that identifies possible issues, and in each of those cases someone must review the data—which typically is not a problem; for example, a false positive. To reduce this workload, we applied machine learning to train a model based on historical data—using the credit files plus human evaluations about their “bad/good” labeling. In production, the model reduces the rate of false positives by 80%, reducing the number of files that require human evaluation. Overall, the AI system combined both machine learning and human insights to make the solution much more efficient.

Take time to understand what your project needs before beginning to build a solution. That might be an iterative process: it might require that your data science team explores the data first, maybe developing a proof-of-concept, and then your organization decides what outcomes are needed. After you have a sense of what kind of work your project will require, take an inventory of the skills on your team.

A good way to visualize this is to build a matrix of the people on the team versus the skills needed for the project, such as the one depicted in [Figure 3-2](#).

Your project needs might require different columns. For example, the Production Systems column likely includes data engineering and platform support, although in some projects there will be significant need for application developers to integrate machine learning

components into production systems. Perhaps your organization has a separate data engineering team, but requires that you embed application developers to work side by side with data scientists. For another example, in some verticals, such as finance, compliance requirements and other domain expertise can become quite complex, so you might need Domain Expertise as a column. Adjust the matrix to suit your project needs.

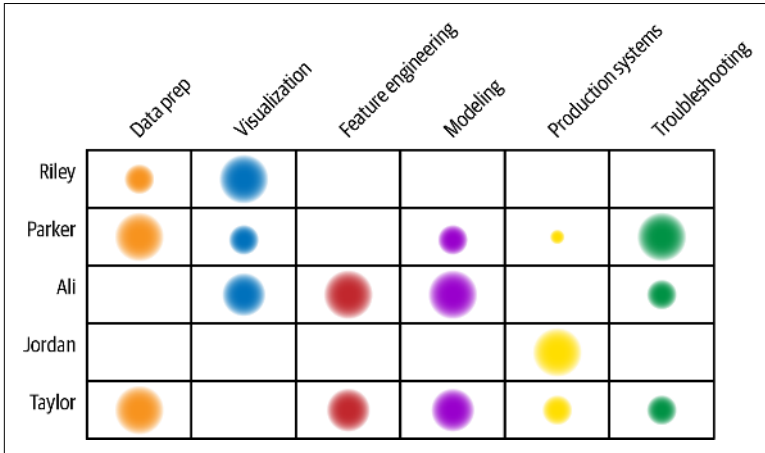


Figure 3-2. Matrix to assess team skills versus project needs

Some members of your team might be proficient in multiple skills (e.g., a data scientist who previously worked in DevOps), whereas other people might be more specialized. After you populate that matrix, a gap analysis should be clear. What areas warrant more training for your team? Are there people and skills missing from the team? These are the kinds of questions to begin assessing after you have that gap analysis in hand. Again, it's about finding a balance of complementary abilities to build a robust, interdisciplinary team.

## Core Skills

Let's look through the core skills for data science, considering each potential stage of a project.

## Data Preparation

Much of the work performed by data science teams focuses on cleaning and preparing data. When an organization uses machine learning, core business value creation comes through a well-known set of activities: continually improving data sources; resolving data quality issues; finding better means for feature engineering; digging into the edge cases with data that might introduce risks through bias, privacy, security, or ethics issues; and so on.

By analogy, if you worked at an investment fund where most business value creation came from carefully managing the portfolio of investments, most of the time spent by your organization would probably be focused on curating that portfolio. Data is a similar game. It would be impossible for a business to get all of those data sources 100% correct on the first try. Instead, much of that work depends on interacting with customers and vendors and arriving at a better understanding of the data and use cases over time—similar to managing a portfolio of investments. It makes sense that so much data science activity is focused on data preparation.

Important characteristics for *data preparation*, which should be essential for almost everyone on a data science team, include the following:

- Curiosity. Improving the data is what brings value, and being curious is one of the best ways to approach that.
- Data wrangling skills, such as using SQL queries and pandas or other popular libraries to get your data into the needed shape.
- Understanding how to navigate through different kinds of data management frameworks.
- Some general programming skills: Python coding, Bash scripting, debugging, and so on.

Data management skills such as SQL queries are generally needed to pull data. Some kinds of data might not fit well into relational databases and instead need to be pulled from key–value stores, column-wide stores, graph databases, pub–sub, and so forth. SQL queries alone will go only so far. Learning to work with a variety of data frameworks is important for anyone on a data science team.

## Coding Chops

Overall, data preparation is a process that needs to be repeatable, so programming skills are vital. Programming languages such as Python and R have become hugely popular for data science work. Some of the big data distributed frameworks can be implemented in Java or Scala (such as Apache Spark) or in C++ (such as TensorFlow), so having familiarity with a range of programming languages can be important. Not everyone on a data science team needs to be fluent with every programming language needed; this reinforces the need for skill diversity.

Along with the programming languages, adjacent skills in versioning (Git), debugging and issue tracking, performance analysis, and other troubleshooting are important too. Probably the best skill is to know how to find answers when you become stuck on a programming problem by pairing with other people on your team, searching online, or researching previous work.

Most important, developing processes so that data preparation workflows can become reproducible is critical for the “science” aspect of data science. Using tools such as Jupyter Notebook is a great way to help make data preparation workflows reproducible while also helping make results more understandable by stakeholders. Collaboration based on Jupyter often implies running atop Docker containers and Kubernetes clusters as well as other system level skills such as Bash shell commands and other Linux use.

## Exploring and Visualizing Data

Now that you have your data in shape to use, the *discovery stage* is where you explore it to find patterns and trends. That can involve statistical analysis, and it might need some machine learning (such as clustering). If the end goal is to produce reporting for stakeholders, the job will require some domain expertise about the business. People working at this stage need good communication skills and should work closely with stakeholders to tell the stories within the data. Writing reports and articles, public speaking, developing interactive explorations of data insights through Jupyter Notebook—these are all vital communications skills on data science teams.

*Data visualization*—that is, composing a story visually with data—is an effective and compelling way to convey what your team discovers

in the data. The business concerns addressed by data science work typically involve multidimensional problems. For example, an anti-fraud model might need to limit credit card charge-backs, while *also* considering the customer support costs for false positives, guarding against bias toward particular subgroups of customers, and minimizing computational infrastructure costs. Visualization is a great way to explore multidimensional data, particularly for exploring the complex “performance” measures of machine learning models. Stakeholders might be able to spot effects in the data by looking at visualizations, over and beyond what the data science team recognizes. The skill sets involved in data visualization include using libraries, data storytelling, and frontend programming skills.

Many data science projects don’t need to go beyond this point, after stakeholders gain the insights needed. It can help to have people who understand how stakeholders will use the inputs from the data science team. For example, having familiarity with decision science is one way to bridge between data science work and executive decision-making.

## Data Storytelling

*Data storytelling* is a way to communicate effectively with data. There are many tools to help generate charts from data, although those tools alone aren’t enough. The trouble is that people struggle with understanding a collection of facts: clutter, attention span, high dimensionality, cognitive overload—these aspects interfere. Instead, people tend to learn through stories. Although it’s relatively easy to generate a complex chart from data, it’s also easy to generate charts that people will not readily grasp—or worse, ones that lead to misconceptions.

In data science, we talk about how much time is dedicated to data preparation, and, of course, much effort goes into the analysis to reveal insights. However, the “last mile” of converting insights into actions is the most crucial step. By creating a compelling story, stakeholders will recognize what actions are needed. Anticipate common questions and address those at the outset in your presentation. Better yet, create elements of narrative from your analysis, then work side by side with stakeholders to help them craft the story to be told.

For more about data storytelling, we recommend the following:

- *Storytelling with Data: A Data Visualization Guide for Business Professionals* by Cole Nussbaumer Knaflic (O'Reilly)
- The collected works of Edward Tufte, available through Graphic Press LLC

## Platform Engineering

Whether your team is working on data preparation, discovery, data visualization, modeling, or other areas, working with data requires lots of computing resources. Some aspects are compute-intensive, others demand lots of memory, still others need network bandwidth, and there's always the issue of data storage. Making efficient use of computing resources is crucial for the team to be effective. Some organizations call this kind of work *platform engineering*. Others categorize it as site reliability engineering (SRE), DevOps, or data engineering. In any case, you'll probably be using some distributed systems such as Apache Spark or TensorFlow, perhaps some cloud services, perhaps Docker and Kubernetes. Having people with engineering skills to use these systems and develop platform capabilities for the rest of the team is essential.

## Feature Engineering

Machine learning models need training data, at least for supervised learning, which is most commonly used. Data preparation will handle most of the work to develop training sets, but there are important nuances. Machine learning models build upon features in the data. Your project might need to iterate on feature engineering to train effective models, not simply take the raw data. For example, some columns of a dataset might have imbalanced classes such as fewer rows about people from a minority group. If not handled appropriately, that situation can lead to bias in the resulting models.

Understanding feature engineering is an art, one that's loaded with math and science. To dive deeper into this, we recommend reading *Feature Engineering for Machine Learning*, by Amanda Casari and Alice Zhang (O'Reilly). Also check out the open source [AIF360](#) toolkit for a sophisticated set of tools that can be integrated into your feature engineering workflows. AIF360 and tools like it help detect potential bias and prepare data in ways that lead toward better AI trust overall.



## Modeling

Finally! By many media accounts, data scientists spend most of their time being concerned with exotic algorithms. Realistically, that's only a tiny part of the job. It's important nonetheless, and you'll need people who understand the algorithms used to train machine learning models, especially when it comes to evaluating results of machine learning models, given that each family of algorithms tends to use different metrics and evaluation criteria.

In the era of deep learning, modeling has become substantially more costly to compute. Relatively few organizations train deep learning models entirely from scratch; instead, they reuse models that some other organization had previously trained. In the latter case, we use *transfer learning* to adapt a previously trained model, “fine-tuning” the model with data specific to a new use case. Consequently, it becomes important to manage the *hyperparameters*—configurations that control and adjust models—which might number in the thousands, with recent research efforts trending toward *billions*. Hyperparameter optimization can lead to large expenses if not handled well. Using popular frameworks such as TensorFlow, PyTorch, Ray, and others is quite helpful, but you'll need people who have these skills.

## Model Integration and Deployment

Deploying models in a production environment is emerging as its own field: *MLOps*. Many kinds of problems emerge in production only after the models begin interacting with live customer data. For example, if a model's accuracy suddenly jumps, that might be a very bad thing, possibly indicating outliers or model drift. Security, privacy, bias, ethics, and a growing variety of compliance issues emerge precisely at this point of model deployment. These present operations issues that are vastly different from what IT operations staff have had to handle previously.

To build trust for the decisions AI is making on behalf of your business, you need an appropriate strategy in place for model management. This means refreshing models on a regular basis, monitoring accuracy of predictions, and other feedback loops such as capabilities to explain a model's decisions. Model trust is especially important in regulated industries to make sure that the automated

decisions are not biased and that they properly represent your brand.

## Troubleshooting in Production

Over and beyond what the MLOps team will need to handle, it will be required to do some troubleshooting in the live production environment. Understanding the edge cases for security, privacy, ethics, bias, and a range of compliance requirements is difficult. It will probably require the people on your team with the deepest experience in statistics, plus lots of hands-on experience with machine learning.

This kind of troubleshooting work, and the risks associated with it, are quite different from other areas of software engineering. Few product managers, so far, have much experience directing projects that depend on machine learning models; these are probabilistic systems and require very different skills and processes than, say, web app development. To learn more about how product management for AI differs from previous practices, see [“Why Machine-Learned Models Crash and Burn in Production and What to Do about It”](#) by David Talby, and [“Why Managing Machines Is Harder Than You Think”](#) by Pete Skomoroch.

## Building Teams

Engineering teams tend to focus on process: how to use or develop APIs most effectively, troubleshooting systems, estimating efforts through time-boxing, paying down technical debt, and so on. All of these are important for developing software, but those priorities reflect the software-deployment end goal of iterating on a code base.

In contrast, data science work encompasses a much broader scope than code and software deployment: organizational decisions must be made, and risks must be considered, all in the context of domain expertise. Instead of coding, data science teams emphasize systems *learning*: specifically, preparing and curating data so that it engenders learning, produces insights, and augments decisions.

## Team Culture

The culture that emerges within data science teams will be different from that in engineering teams. Their priorities and risks are

different, so their processes and methods are different. Whereas software engineering teams many times align closer to technology, data science teams align closer to business. Data science teams must evaluate risks; they typically work with probabilistic systems, unlike the deterministic systems that software engineers use. The results from data science teams help to drive business outcomes, where executives are focused on decision-making processes.

Think about how a software engineering team builds a web app. Typically, the most experienced members of a team work on early stages, such as requirements, API design, architecture, and so on. As the project matures, less-experienced members of the team develop individual features, unit tests, and so on.

In machine learning work this scenario is reversed. Building a machine learning model from a given dataset is something even the least experienced member of a data science team should be able to do. However, after machine learning models are deployed in production, unanticipated problems emerge: fairness and bias issues, security issues, privacy, model drift, and edge cases that require advanced statistics to troubleshoot. Those problems require the most experienced people on the team. They also tend to involve compliance and regulators, so mistakes at that stage can escalate to the executive level. In other words, the life cycle and process for machine learning in production are nearly the opposite of what's needed for web app development. It's no wonder that a substantially different culture emerges for data science teams.

Even though engineering teams typically take requirements from product managers and execute on those, data science teams are often involved in more exploratory work. A kind of “startup mindset” comes into play: the data science team might need to perform a lot of work at the outset to see whether an idea would even be feasible with the data available.

## Finding the Best People

You have the project analysis in hand, and you've run a gap analysis on the skills your team already has versus what your project needs. How do you find the appropriate data science professionals?

Computer scientists don't necessarily have these skills. They'll generally need training before they can contribute to a data science team. HR departments might advertise for “PhD or better in computer

science” on the job posts, but that’s not what your data science team needs.

Oddly enough, scientists are often a very good candidate pool. Early in the history of data science teams in Silicon Valley firms, it became apparent that roughly half of the people hired had come from physics or physical science backgrounds. Physics grad students collect and prepare data and run analysis, visualizations, and modeling, as well learn to use a wide range of distributed computing platforms. There also tend to be many more physics grad students than there are jobs open for them. So that’s a good way to hire people familiar with the tools and processes needed in data science teams—people who can be productive on day one.

Although physics tends to utilize machine learning and high-performance computing, social science research tends to engage with statistics, follow protocols for ethical handling of data, and anticipate outcomes on the people involved. The latter tends to use smaller datasets, but the data typically concerns confidential information about human subjects: wages, medical history, family background, home addresses, and so on. Privacy concerns, biases, ethics, security, and data governance—these issues, which are innate practices for social science research, are rapidly becoming top priorities for data science teams. Moreover, social science researchers generally understand how to study human behavior, which is often where data science teams in industry need to focus: on customers. Perhaps even more to the point, and similar to physics grad students, there are generally many more social science grad students than there are jobs open for them.

There are many good programs for people interested in “upskilling” into data science roles. One point to keep in mind: even though some candidates come from degree programs that emphasize the skills, tooling, and mindset needed for data science work, there are still relatively few university programs that focus specifically on this interdisciplinary field. Also, the field is evolving too rapidly for any one individual to acquire all of the skills needed; in other words, data science teams require continual learning and good, ongoing mentoring programs to be effective. Therefore, plan to provide training. Foster a culture for ongoing mentoring because there’s much that your people will need to learn, especially from one another. Build paths for internal hires; that is, people who want to “reskill” to move into the field. You’ll get people who are already

familiar with your business and bring extra domain expertise into your data science team.

One area that is not as easy to capture is the culture of your team. In our experience, a resume is half the story; the other half is having the appropriate mindset. A successful team generates endless curiosity and passion: the curiosity to understand problems and make things better, and the passion in the practice of AI and to drive change and make an impact. Data science work is not easily prescribed, and every company will face different challenges. We like to think that each company must develop its own unique data science practice for its lines of business. Hiring people with the proper mindset becomes crucial to support that. You also must hire to diversify your team's mindset. Avoid building a team where everyone has the same background. Also, less-experienced data science team members who bring the aforementioned mindset can be unleashed on business problems and really make a big impact on your business. Build a team of people who are empowered to make a difference.

Overall, try to get people who cultivate curiosity and are good communicators. If needed, you might even organize a Toastmasters group for your team to practice their public speaking skills.

**NOTE**

We recently helped host an MLOps Day track at OSCON 2019. (See [MLOps Day, OSCON 2019](#) for discussion about the talks in that track, including slides and videos.) The speakers presented their experiences with production uses cases of machine learning in the enterprise, including teams from Capital One, Comcast, GitHub, Red Hat, and more. For other good examples, see [“Building a Rock Star Data Science Team: My First Year with the IBM Data Science Elite Team”](#) by Carlo Appugliese.

# Summary

As you can see, artificial intelligence (AI) has many moving parts. A practical Agile approach to AI focuses on the right team mindset of flexibility, the right set of tools, and the right set of team skills. Across the first three chapters, we talked broadly about those ideas as well as identifying the proper uses cases, organizing your data, and changing your business by teaching it to trust the products you're building. We want to take a few last words to close on the significance of these points so that you can go forth leading your team to achieve its most ambitious AI projects.

## Use Cases

Integrating AI into your business happens one use case at a time. First, work to identify the appropriate use cases: use workshops and design thinking, and ideate as a team to get many perspectives and insights. Focus on business problems; don't get caught in the trap of building technology for the sake of technology.

After a project, conduct retrospectives on those use case implementations. What worked well? What could've been handled better? Identify patterns in your use cases, and share those across the organization. Ideally, create a Center of Excellence for AI patterns within your organization.

# Data

You can't deploy AI without first getting your data into shape. Those are table stakes, the absolute basics. Recent **industry surveys about AI adoption in the enterprise** have shown that the majority of firms become bogged down in the technical debt they must resolve before they can share data across divisions. Also recognize that data sharing runs counter to the inertia that tends to arise in large organizations. When people have complex tasks to manage, it's generally simplest to break those into smaller, simpler tasks and then keep the parts separate. Organizations respond to complexity in much the same way over time, establishing silos between divisions. That's especially true when it comes to data management and access controls: the easiest answer to a request for data access is "no." However, the easiest answer is not always the best in the long run.

Some people have called data "the new oil"—an economic good driving enormous value—but it's better to think of data as an investment portfolio. Your data science teams help manage those investments in data, and they help realize equity and yield from them.

One idea that's been making the rounds is *data democratization*: an idea in which nearly everyone in a company can run queries, create analysis, generate reports, and so on. Obviously, not all the data in a firm will be made available to every employee, given that there will likely be confidentiality concerns to balance, as well. Data democratization has drawbacks: making claims about data analysis depends on having enough of a statistics background to get it right—although this is where data science teams can help coach and amplify results from the rest of the organization.

There's a related term to data democratization: *citizen data science*. This is when people throughout your organization become involved in developing data insights; if you think of data science as a continuum in which people on one end are just beginning to "upskill" into these kinds of roles, and experts on the opposite end are continually learning new skills for their data science practice, citizen data science makes a lot of sense for a large organization. Drawbacks and critiques aside, both of these approaches can make a lot of inroads toward breaking down data silos.

## Tools and Process

To get a job done well, you must have the appropriate tools and apply the appropriate process to them. Embracing open source is an excellent first step because there's so much available and robust developer communities to learn from. Also, don't limit which tools a data science team can use: let it evaluate alternatives and use what's appropriate for the people and the challenges involved. Bring in new tools to support new talent while utilizing the existing approaches and tooling to support your existing talent. For example, some of your staff will need to use Python and machine learning frameworks, whereas other people might be more focused on SQL and reporting.

Be agile in your approach! This means keeping in mind that effective work is not about building the best possible model for one use case; it's about the cadence you establish for creating many effective models for a range of use cases. Invest time to prove value; however, time is typically the most expensive resource in business, so don't spend too long to prove value in a given use case. Time is not your friend. Joni Rolenoitis, Experian CDO, said it best, "Go agile or go home."

## Mindset

You need to foster the proper kind of culture for a data science team so that the members can develop an effective mindset for the challenges they face. Data and dogma don't mix well. Data science teams are often in the "hot seat" because they're the gatekeepers for analysis that disproves what others in the organization might take as givens. On a data science team, we're fostering an analytic mindset, one that's driven by curiosity. To support that, create an environment in which people can collaborate with their peers, where criticisms coming from outside the team are constructive, and in which opinions—both inside and outside the team—are respected but not worshipped. Ultimately, data and reproducible analysis, and an extra dose of curiosity, must rule over opinion.

Diversity helps. Diversity builds a wide range of skills and experience across a team, which combine to produce effective solutions. Data science is inherently interdisciplinary. Diversity also helps temper opinions so that they don't turn into bias.



In terms of organization and planning, some aspects are crucial for establishing an effective mindset:

- Keep your sprints quick when you need to prove or disprove value, but give your people time to invest in a problem that needs to be resolved.
- Develop team deployment models that fit your business needs. Some data science practices use a hub-and-spoke structure to get their members involved in use cases while still closely communicating. In other firms, there's an "embedded" model of rotating people into product teams, but bringing them back together periodically to compare notes. Experiment with what's best for your organization.

## Integration and Trust

Integrating AI into your business will probably have at least two dimensions. First, there's the impact on your internal business workflows. How does the organization arrive at decisions? How should automation affect the decisions that people make? Second, there's the decision of where AI fits into your roadmap for customer solutions. Develop your analytics and machine learning work with these end goals in mind: how will they be consumed?

In either of those categories—impacting internal operations and product decisions—trust is key. To build trust in AI, people need to be comfortable with automation systems and confident that they produce good results. Good results go beyond the accuracy of predictions; your organization and customers must also be comfortable with your data science team's practices addressing concerns about fairness and bias, developing models and supporting workflows in which decisions can be explained and explored, and designing for security and robustness. There's a growing body of open source projects to help remediate concerns about AI trust, and related foundations such as [LFAI](#); for example, the Linux Foundation's [AIF360 toolkit](#) provides a range of advanced statistical tooling for measuring and correcting bias in training data.

# Conclusion

To close this book and pull together the components described throughout it, your challenge is to build a data science strategy for your organization. As **Figure 4-1** shows, we approach this challenge from two directions. We want a culture that supports discovery, diversity of skills and perspectives; embraces new ideas and insights about your business; and moves quickly to prove or disprove them. We also want an effective data platform that supports a data science team that has diverse needs, along with scaling for customer use cases.

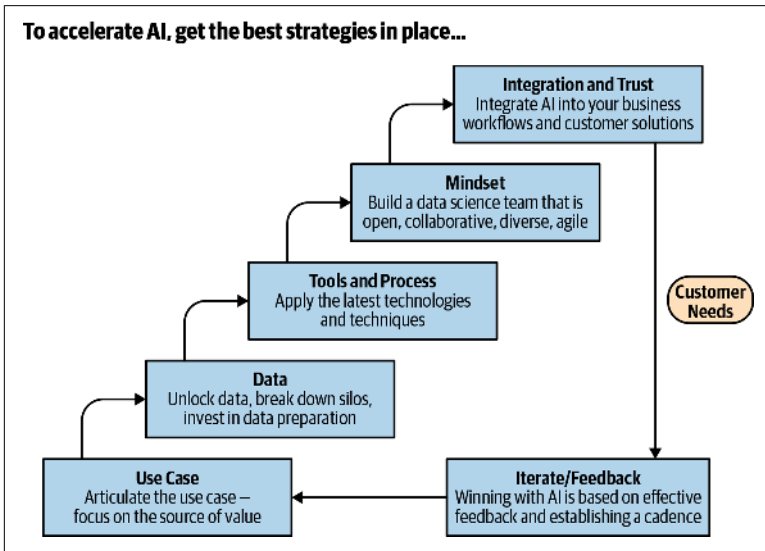


Figure 4-1. Strategies for accelerating AI

The upper-right corner of **Figure 4-1** faces closest to your customers. AI trust is paramount there, and to reach that, you must build an appropriate culture for your data science team and for their relationship with the broader organization. Your data science team needs the skills, the diversity, the mindset, the mentoring, the curiosity, the domain expertise, and continuous learning to take advantage of machine learning and integrate data products to meet your customers' needs.

The lower-left corner of the diagram is about your data and investing in its value. Break down the silos, collaborate across teams, and get the appropriate tools and processes in place to unlock the value

of your data. Make these platform capabilities meet the needs of your data science team's culture and skills, both for immediate purposes and as their curiosity and insights and organization learning grows. Also, make your platform capabilities meet your customers needs as your business grows.

Ultimately, these two approaches must meet in the middle. Throughout this entire process, there are feedback loops that you can recognize and nurture so that both your platform and your team improve continuously. With these practices, skills, and culture together, you can be agile in building AI in your organization.

## About the Authors

---

**Carlo Appugliese** is chief AI evangelist at IBM within the data and AI product development team. Carlo is the founding member and leader of IBM's Data Science and AI Elite team. The Data Science and AI team is a team of highly skilled data scientists and engineers responsible for helping IBM clients leverage AI. Working on hundreds of projects, Carlo and the team have developed an Agile methodology to quickly prove business value with AI. Carlo has built his career leading highly technical teams leveraging emerging technologies and trends to break conventional norms to drive rapid business innovation. Carlo has previously served as a software engineer, application development manager, and director of innovation.

**Paco Nathan** is known as a “player/coach,” with core expertise in data science, natural language processing, machine learning, and cloud computing. He has 35+ years of tech industry experience, ranging from Bell Labs to early-stage startups. He is cochair of the Rev conference for Amplify Partners, Recognai, Primer, and Data Spartan. Recent roles include director of the learning group at O'Reilly Media and director of community evangelism at Databricks and Apache Spark. He was cited in 2015 as one of the Top 30 People in Big Data and Analytics by Innovation Enterprise.

**William S. Roberts** is a senior data science evangelist at IBM within the data science and AI technical marketing group. He seeks out the most compelling stories of IBM's data science practitioners and shares those success stories with the larger IBM Data Science community. Prior to IBM, William worked at Red Hat Inc. as a middleware consultant, helping financial clientele modernize and automate their business processes through application development. A Bay Area native, William currently lives in San Francisco.