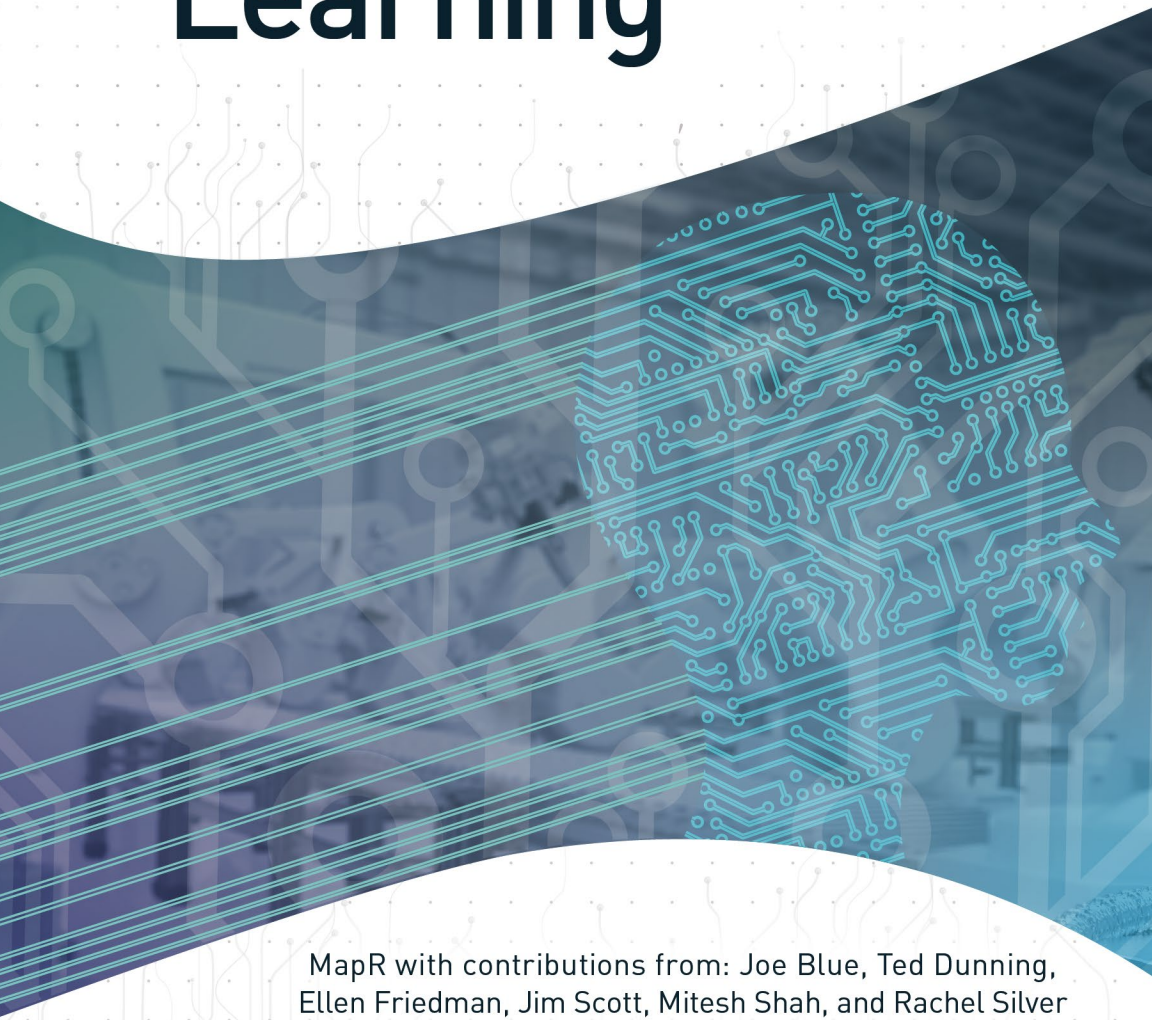
COMPLIMENTS OF
MAPR

EBOOK

Buyer's Guide to AI and Machine Learning



MapR with contributions from: Joe Blue, Ted Dunning, Ellen Friedman, Jim Scott, Mitesh Shah, and Rachel Silver

Buyer's Guide to AI and Machine Learning
MapR with contributions from: Joe Blue, Ted Dunning, Ellen Friedman, Jim Scott, Mitesh Shah, and Rachel Silver

Printed in the United States of America
Published by MapR Technologies, Inc.
4555 Great America Parkway, Suite 201
Santa Clara, CA 95054

January 2019: First Edition

Table of Contents

Transformation Drivers for AI

It took 60 years, but artificial intelligence is finally real.

Ever since the night a Univac computer correctly predicted the results of the U.S. presidential election in 1952, people have dreamed of the day when a machine would be capable of mimicking human thought. At times it seemed we were close. Although enormous progress has been made quickly in getting computers to solve problems that seemingly require intelligence, our understanding of what is really required for intelligence has progressed as fast, and sometimes faster. The result has been that as often as the goal seemed within reach, it has slipped away, not due to lack of effort or progress, but because our perception of what the goal really is has changed.

But all that is changing quickly. An IBM computer recently matched wits with two accomplished human debaters and was judged the winner. Google has demonstrated a feature of its Assistant that can make restaurant or salon reservations by voice while deftly handling such anomalies as accents and misunderstandings. Telemarketing robots can now engage prospective customers in convincingly real sales calls and computers are writing news articles about corporate earnings and sporting events that are almost indistinguishable from those created by human authors.

Is this artificial intelligence? Experts have debated the definition of AI, since computer scientist John McCarthy first coined the term in 1956. The threshold has often been the Turing test, which is the ability of a computer to exhibit behavior that would be indistinguishable from that of a human being. However, that kind of interaction is only one small part of the AI landscape.

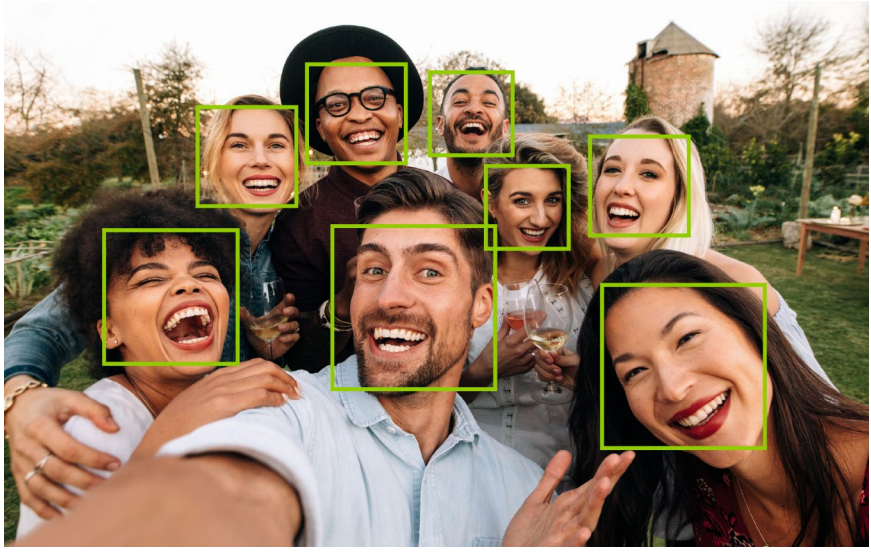


Figure 1. Face recognition is just one way in which image recognition is being put to use.

Today, AI is being made real in applications like recommendation engines, chat bots, image recognition, and robotics. Computers are already better at identifying objects and faces in photographs than are humans in many cases. The ability of computer translation of human language has greatly improved, although it still falls well short of what human translators can do. AI applications can identify the likelihood of adverse drug interactions more accurately than many human doctors.

Rather than asking, "Are these examples of artificial intelligence?" the better question might be, "Does it matter?" Regardless of what you call it, computers today can tackle tasks that would have been unimaginable just a few years ago.

Research firm Gartner defines different types of analytics in ascending order of difficulty: descriptive, diagnostic, predictive, and prescriptive. Descriptive and diagnostic analytics look at historical data to try to figure out what has happened or why something happened, such as the failure of a machine. Predictive and prescriptive analytics are more forward-looking, using patterns identified in historical data to figure out what will happen in the future and recommending actions. All of these kinds of analytics can benefit from techniques such as artificial intelligence and machine learning, but predictive and prescriptive analytics are often only possible using AI and machine learning.

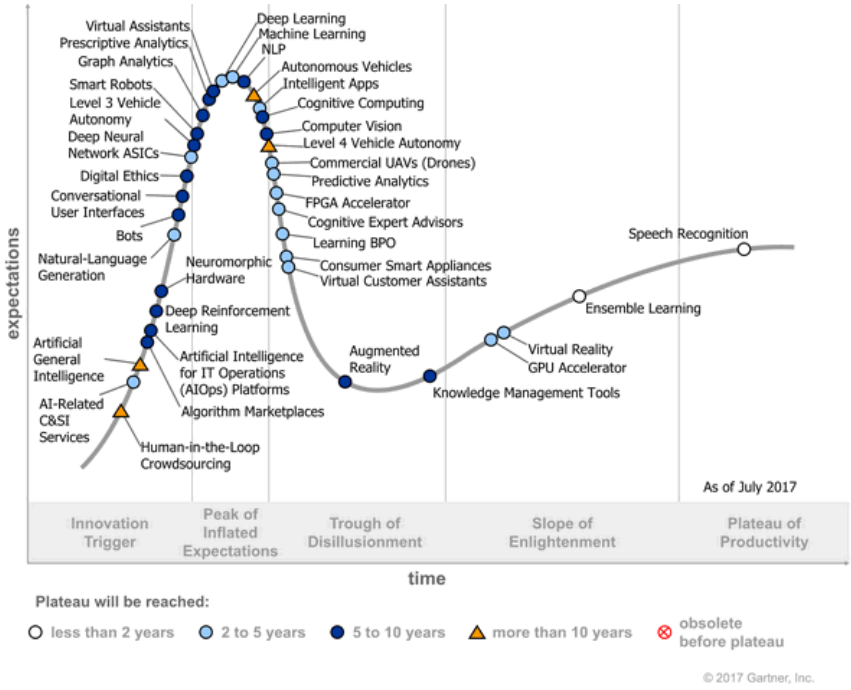


Figure 2. This Gartner report shows an evolution of expectations for AI and related concepts including forms of machine learning and analytics. Keep in mind that the terms "AI" and "machine learning" do not represent cleanly separated approaches. (Image: Gartner 2017 AI hype cycle).

Developers and data scientists often focus on the tools, libraries, and frameworks needed to get their jobs done. Any good craftsman should know their tools and keep them sharp, but the choice of tool or skill with it matters little without seamless, secure access to the right data. Equally important is the ability to deploy models efficiently. AI and machine learning applications are worthless until you deploy them and have them start making decisions. These needs are why the capabilities of the data platform are a critical part of success for AI.

Traditional relational database management platforms are poorly suited in many ways for most AI and ML applications as shown by the examples we will describe in this book. In particular, analytics and AI both require access to accurate historical data for learning, troubleshooting, and forecasting. Traditional databases are not designed to preserve the past exactly; they are meant to be overwritten. They show you the current state of your data but cannot necessarily show you past states without

special application coding, and they are inefficient for dealing with really large-scale time-series data. Choosing a platform that supports immutable persistence, streaming message transport, and industry standard access methods for files and other data sources is critical to giving your data scientists and developers the flexibility of a full range of tools for their work. You'll find more detail in Chapter 3.

Part of what drives the trend toward AI is the larger recognition of the value of data, particularly large-scale data, and the recognition of how data-driven decisions can transform business. Done well, the impact of digital transformation can be dramatic. For example, Netflix carries no sales costs on its income statement because it has no salespeople. The company makes it drop-dead simple for customers to provision their own accounts. Netflix has just three price tiers, which customers can choose and change as they please. The company's famous recommendation engine uses machine learning to suggest content it thinks subscribers might find valuable while also helping Netflix make programming decisions. Customers do not interact with human employees of Netflix as they use the service.

As a result, Netflix's general and administrative expenses are just 0.7% of revenues, compared to 20% for a conventional retailer and nearly 50% at Blockbuster in 2010, which was the year Netflix effectively put Blockbuster out of business. Netflix continually harvests data about subscriber behavior to identify concepts for original programming. Netflix is able to spend more than $8 billion annually on original content in part because it spends so little on everything else.

Algorithmically driven businesses like Netflix, Google, and Amazon have vanquished competitors because of vast efficiencies created by automation and machine learning. And they're only getting started. David Moschella, a prominent futurist and research director for CSC's Leading Edge Forum, has suggested these companies are building versatile "digital fabrics" that will easily carry them into new industries. For example, Amazon's forays into insurance and healthcare are supported by the same kinds of predictive algorithms it uses to recommend music and appliances. Competitors who can't match their cost structure and superior user experience will struggle to survive. Regardless of whether you call these software enabling these business "intelligent," that software is redefining the ways entire industries operate.

But you don't need to transform an industry to realize benefits of AI and analytics. Finding points in your business process where automation can open up a bottleneck or improve customer experience is a good way to figure out where AI may help. And for AI to be of value, it doesn't always have to be a complex solution. A relatively simple example is customer segmentation based upon factors like demographics and buying history. There are many tools that can enable businesses of any size to better understand their customers and target promotions accordingly, realizing significant cost savings.

Off-the-shelf tools like chat bots and intelligent email response systems are affordable and can deliver big productivity benefits by engaging customers before they bounce off a website or by delivering drip messages tailored and timed to the needs of individual customers. Intrusion detection systems powered by analytics can head off cyberattacks, the average cost of which now exceeds $3 million per attack. While none of these applications disrupt industries, they save costs, improve sales, and reduce risk, which ultimately helps organizations stay competitive in an increasingly cutthroat business environment.

Business Value Drives Adoption

Not only have methods in AI evolved, but so has recognition of the ways in which organizations can leverage these techniques for real business value. Early uses of AI centered on data mining, predictive analytics, pattern recognition, and recommendation engines. These are essentially evolutionary steps from big data analytics, with machines churning through troves of information too large for humans to process in order to identify relationships.

Regardless of whether these applications are technically AI, they're useful in many contexts. For example, many consumer companies process an enormous number of support requests coming in from multiple channels such as email, social networks, and their own websites. The task of sorting through support tickets is time intensive for humans, but machine learning algorithms can quickly identify word patterns and classify tickets into the appropriate categories. The algorithm can then route the ticket directly to a service agent without the need for a support person[1]. Machines can even answer some basic

1 https://mapr.com/blog/how-machine-learning-revolutionizing-digital/

requests in conversational language, taking that burden off the human part of the support organization.

Computerized human language translation has made big progress, and computerized translation of text, such as webpage translation, is sufficiently accurate to be useful in many business settings. Image-based translation, including phone-based applications, also has come of age. You can point your mobile phone at text written in one language and read it in another.

Predictive maintenance analytics algorithms have been used for some time to help identify equipment that is at high risk of failure so it can be serviced without downtime and without potentially catastrophic damage. An evolution of that in the oil and gas industry uses machine learning to analyze data from thousands of sensors on wellheads to predict the existence and volume of underground supplies, thereby dramatically reducing exploration costs[2].

As a final example, machine learning is emerging as an important ally for increasingly needed skilled cybersecurity professionals. While keeping bad actors entirely out may be all but impossible, AI can help minimize the damage they inflict. Machine learning algorithms can scour systems or network and database access logs to detect patterns that indicate anomalous behavior at a scale that's beyond the reach of human operators.

Why Now?

As with most technological leaps forward, AI's recent dramatic progress is due to a combination of improvements in hardware and software combined with cloud deployment methods, the availability of *much* more data, and the overall recognition that approaches such as artificial intelligence and machine learning are not only within reach but also offer big rewards. In fact, one of the biggest drivers is simply the increasing awareness that AI and machine learning are providing value across a wide range of business sectors. Seeing someone else's success is a motivating force to figure out how take advantage of these techniques in your own business.

2 https://mapr.com/blog/five-data-driven-ways-for-oil-gas-companies-to-beat-volatility-oil-prices/

Big data is one of the most important factors in AI's rise. Machine learning and AI applications can require data sets of both current and large amounts of historical information. Training in these systems involves continually creating new versions of models, with ongoing evaluation and modification of models particularly to respond to changing conditions. The most advanced modern machine learning systems require massive amounts of data. Machine learning and AI have been done previously, but storage and processing of information on this scale was for many organizations prohibitively expensive. The advent of big data platforms is democratizing AI and machine learning through innovations like highly scalable clusters with cost-effective storage and good data management. Other advances such as stream transport that couples performance with persistence at scale have made large-scale data systems easier to build and deploy. This makes it easier to collect data for machine learning as well as to deploy the resulting models into working systems.

Quantity of data is just one issue for AI, however. Also important is data access, particularly of comprehensive datasets, and avoiding data silos that may lead to skewed results. In modern businesses, developers and analysts often need to access data from a variety of sources and across geographical locations or between on-premises data centers and cloud deployments. All this needs to be taken into account for effective production systems. Fortunately, advances in data and application management software are making this feasible.

Figure 3. Machine learning frameworks.

Besides the data, new AI and machine learning tools are making it easier for a broad range of organizations to build AI and machine learning systems. For example, machine learning frameworks—such as Google's TensorFlow, Microsoft's Azure ML Studio and the open-source scikit learn and H2O—ease the process of acquiring data, training models, serving predictions, and refining future results. More than two dozen frameworks are available for various languages, programming models,

and use cases, and many are available under open-source licenses. These frameworks can dramatically simplify the process of building AI applications. Advanced techniques like transfer learning enable models built to solve one problem to be re-used as a base to solve other problems, possibly in other domains. This lowers the burden for adoption and means that companies may be able to get started in machine learning without acquiring the massive data assets needed to train a model from scratch. (See Chapters 2 and 3 for more detail.)

Improvements that drive AI now are not just limited to software. Big advances in specialized hardware are also proving to be potent enablers for some types of machine learning. Graphics processing units (GPUs), for example, enable massive improvements in performance especially for applications that involve huge amounts of easily parallelizable numerical processing such as video processing and image recognition. GPUs, which were originally developed for use in computer gaming, are designed to rapidly manipulate and alter memory to speed up image-rendering. They are highly parallel by design, which makes them ideal for the repetitive nature of machine learning processes. While not as flexible as general-purpose CPUs, GPUs are extremely fast for tasks that can be parallelized on a large scale. Using GPUs in the right situations can take the computation and memory-intensive burden off the CPUs, freeing them up for more appropriate tasks such as fetching data to feed to the GPUs.

NVIDIA, which is the world's largest maker of GPUs, has demonstrated a dedicated supercomputer with 16 individual chips – each equipped with 32 GB of memory – working in parallel connected by the company's NVSwitch backplane technology. Many hardware makers are now producing GPU-powered machines specifically for AI applications.

A very recent advance by NVIDIA, the RAPIDS data science framework, is making the use of GPUs become much simpler. RAPIDS is a set of libraries for executing end-to-end data science pipelines completely in GPU memory, benefiting both the training and production cycles. RAPIDS takes the burden off the CPU by moving workloads to the GPU, where the workload will run faster. Jim Scott, VP of Enterprise Architecture at MapR Technologies, says of RAPIDS, "With this release, NVIDIA has taken a major leap forward in simplifying the use of the GPU by doing the hard work for all of us who want to take advantage of the accelerated compute capabilities of the GPU."

Figure 4. With input from the machine learning community and GPU Open Analytics Initiative, RAPIDS is a collection of open-source software libraries designed to accelerate data science and analytics pipelines on GPUs.

The value of CPUs isn't standing still either. Their prices continue to decline while innovations on the horizon like tensor processing units (TPUs) promise to bring new hardware acceleration to bear specifically on AI applications. MapR's Ted Dunning has noted that today's servers are many thousands of times more powerful than the original Cray-1 supercomputer with clustering capabilities that can yield one million times the performance of what was once the fastest computer on the planet at a tiny fraction of the cost.

The combination has flipped the equation of AI development. Previously, computing hardware couldn't keep up with algorithms or store enough data, but processing power and storage space are now relatively cheap problems to solve. The more expensive elements relate to labeling and acquiring data sets.

Another advancement facilitating AI, machine learning, and analytics is the widespread use of containerization for applications. Containers simplify application deployment and help ensure predictable and repeatable environments, which is important for model-to-model evaluation and for reducing the chance of surprises when moving from test to production deployments. Containerization of applications running simultaneously in different environments is key to effective multi-tenancy, and since the environments required for machine learning change quickly, containerization is key. Dealing with the management of containerized applications is an essential part of making this successful. The framework known as Kubernetes, a rising star in the orchestration of containers, is another example of recent advances that are paving the way to better AI and machine learning in large-scale, multi-tenant systems. The pairing of Kubernetes with container technology allows applications to be moved quickly with minimal effort between physical environments, including on-premises, in the cloud, or both.

And that brings us to cloud computing.

Why the Cloud Matters

Cloud computing has played an important role in accelerating AI's popularity by democratizing access to computing resources that were once the domain of only the largest enterprises and academic research labs. Every major cloud platform vendor now offers machine learning services, some even for free on a limited-access basis. This fact has made it possible for anyone to experiment with technologies like machine learning.

The cloud is also an important option for businesses of all sizes to consider when formulating their AI development plans. As noted earlier, AI applications require a lot of processing power and storage, especially during training cycles. Cloud computing can be particularly useful when bursts of heavy compute resources are needed. For more even workloads, on-premises computing may be a better and less expensive choice.

Buying and managing storage devices for on-premises use with very large amounts of data can be somewhat expensive, so low-cost object stores like Amazon S3 are a good alternative in many situations. Similarly, GPU-enabled virtual machines that would be prohibitively expensive to buy for short-term projects are also now available from all major cloud vendors on a per-hour basis. AI developers can tap into powerful GPU-enabled processors running a full suite of development and testing tools at prices of a few dollars per hour using one of the many commercial cloud services. This doesn't mean that use of GPUs is always less expensive via a cloud vendor, however. Usage levels and patterns should be evaluated in each situation, as on-premises GPUs are sometimes more cost-effective if you have sustained workloads. The point is data science teams now have more options to get the compute power they need.

The reasons to consider cloud go far beyond cost savings. Experienced cloud users site factors like business agility, speed of deployment, and availability of sophisticated tools as more important drivers. One recent survey of Amazon Web Services users found that machine learning will be the top catalyst driving cloud computing adoption by 2020.

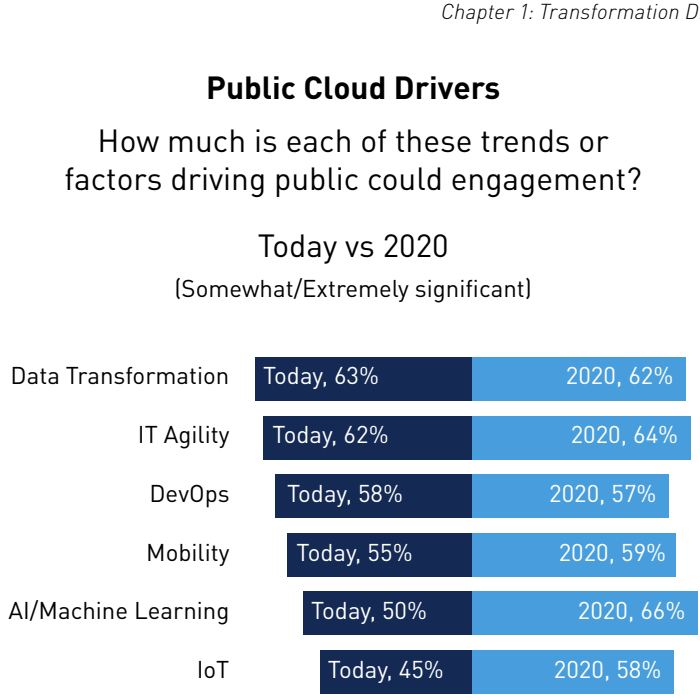


Figure 5. Based on Image
https://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020/#58ee153f6261

Access to development tools isn't the only attraction of the cloud. Many software-as-a-service vendors are now building extensions to their AI applications that permit developers to integrate functionality via APIs. Salesforce.com's Einstein and IBM's Watson are just two examples of such services. Instead of reinventing the wheel, developers can build on top of sophisticated functionality for their own business needs.

There are three basic cloud infrastructure models: public, on-premises (essentially a private cloud), and a combination of those two called a hybrid. Public cloud providers furnish a rich collection of platforms and tools in a virtualized environment that is shared by many customers. On-premises infrastructure can act as a private cloud that is behind a corporate firewall. Organizations may choose an on-premises private cloud when security, compliance, and control over sensitive information is of utmost concern. Hybrid cloud combines both public and on-premises resources to enable companies to shift workloads as needed for the purposes of load-balancing, development/test, performance, and cost control. Hybrid is the dominant model used in enterprises

today, and it is expected to remain so for years to come. Hybrid is not only a useful stepping-stone in moving from entirely on-premises to an entirely cloud infrastructure, hybrid is often an end goal as the best way to optimize resources.

Sophisticated users of public cloud are now increasingly adopting a multi-cloud model and very often using an assortment of public and on-premises resources as well. This gives them the greatest degree of flexibility in shifting workloads around for reasons of performance, cost, and availability of tools. A multi-cloud approach also provides protection against lock-in, which is an increasingly important issue to cloud customers, by forcing cloud vendors into the position of a commodity supplier. Without the right data platform, multi-cloud can result in a difficult to use non-uniform computing environment. We discuss this challenge and several solutions in detail in Chapters 3 and 5. Choosing the right data platform can help by abstracting underlying datastorage. For example, Amazon's popular S3 cloud object storage uses a complex API that makes it difficult for customers to switch to another cloud provider. Organizations benefit, instead, by using a data platform that provides a consistent and standardized set of filesystem operations and APIs on any cloud or on-premises datacenter.

With the arrival of the internet of things (IoT), the role of the cloud is changing. Applications will increasingly need to handle data streams coming in from hundreds or thousands of devices for AI processes like resource optimization and predictive maintenance. This reality is already changing the way applications are designed, including machine learning applications. In many cases it's impractical to transmit massive quantities of machine data to a central processing point for reasons that include cost, immediacy, and the availability of sufficient bandwidth. Decisions often need to be made quickly to prevent a machine from failing or to respond to a buildup of temperature or pressure, for example. In these scenarios, users can't wait for data to be sent to and returned from the cloud or a central processing point, so logic needs to move closer to the endpoint with the cloud serving as an aggregation engine. This distributed processing architecture – which some people have dubbed "fog computing" for its diffuse nature – is inherently a very complex development environment.

Overall the cloud is an excellent place to begin your AI journey even if you don't wind up staying there. A rich assortment of tools is available along with tutorials, pre-built models, sample code, and communities. GPU processing, in particular, can be very costly in the cloud so as the scope of your projects grow, you may want to move certain aspects of your core infrastructure back on-premises. *Even so, don't overlook the option of using a hybrid cloud strategy to get the best of both options*. Choosing a data platform that can access data in both on-premises and cloud infrastructure gives you the greatest flexibility.

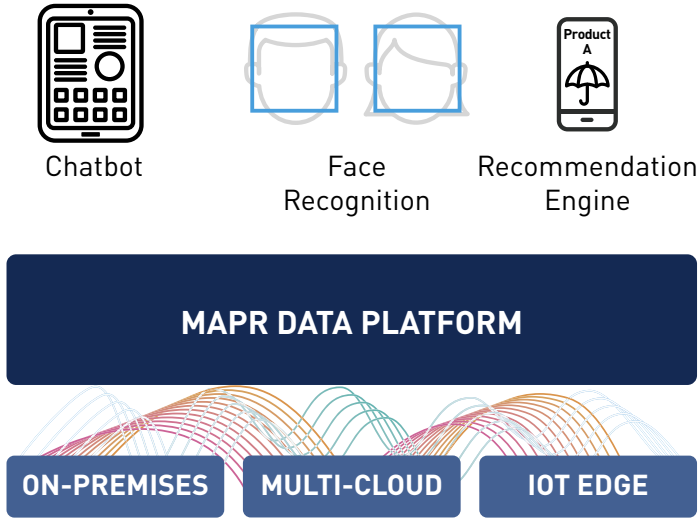


Figure 6. The MapR Data Platform provides a seamless way to access data wherever it is needed, between cloud and on-premises data center out to the IoT edge for collection and processing that is located near sensors.

AI Basics

You don't need to know in detail how to design algorithms, train models, or implement an AI or machine learning system in order to make good choices about what tools and data platform your organization will use or how to tie AI to real business goals, but it is useful to have a basic understanding of how AI works. If you know what's important where data is concerned and are familiar with some basic AI concepts and terms, you'll be better able to build effective data science and DataOps teams to get real value from AI.

A starting point is to realize that building AI/machine learning models isn't really all about advanced math in spite of the impression you might get from online classes on the topic. A lot of it is really just a new way of programming. What's different? It's not a hard-and-fast distinction, but in traditional programming, a programmer starts with an intent, comes up with a specification about each step, and then implements the specification. The intended result is a completed and correct program that is (hopefully) ready for production. Even if it's later debugged or modified, it's essentially done at this point. By contrast, with an AI or machine learning project, the data science developer starts with an intent and makes a plan, but not a plan in which each step toward the goal will be exactly defined and manually implemented. The model will be *learned from historical data rather than written based on a specification*, and once the model is ready, it will make decisions and in a few cases may even continue to learn. The model is, however, a program, just not one we wrote using traditional software engineering methods. That means we are inherently less certain about exactly what a model does than we are with a program we have coded. Of course, this is exactly why we used machine learning in the first place, to solve a problem too difficult to code directly. Models are then evaluated against historical data and ultimately deployed to

production, but there's much more of a sense of not being done – there's ongoing monitoring, evaluation, and a strong presumption that new model versions will be needed. This is partly because we can't be quite sure what the model learned and partly because the model interacts with the world, and the world changes, causing possibly surprising changes in the behavior of the model. AI and machine learning very much tend to be iterative processes that need flexibility and agility to respond appropriately to changes if they are to be valuable in a practical business sense.

Each AI or machine learning project will have specific requirements, but some basic concepts apply to almost all projects. These basic concepts are not only easy to understand, they are important for the non-data-scientist in order to build systems that make sense in terms of resources and that connect to real business value. Figure 7 illustrates some key aspects of machine learning.

Having the right data is essential to success. Note that datasets used for training a model tend to be larger than the input data (new data) used to run the model in production. Computation loads can be heavier during training as well. That's one reason specialized hardware such as GPUs may be more necessary during training than when running models in production.

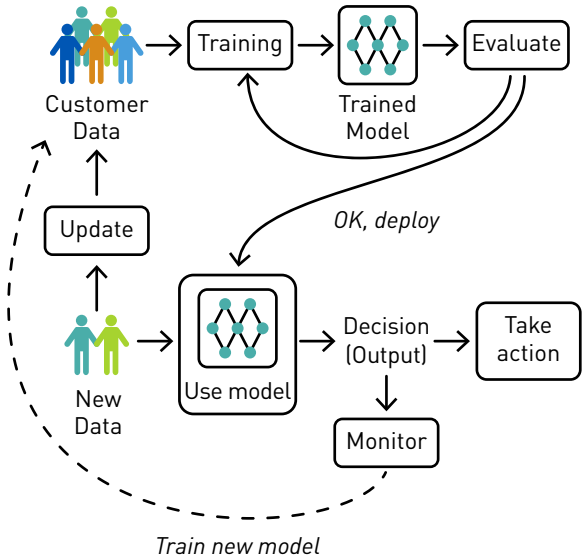


Figure 7. Basic aspects of machine learning. The historical data set used to train models is often much larger than the input data presented to models in production. Evaluation of a newly trained model may indicate the need for tweaking the learning algorithm or the training data and thus a need to train a new version of the model. Even in production, ongoing monitoring may trigger the need to start over and train new models.

In addition, it's important to realize that data and feature extraction are a huge part of what determines success for AI systems and are often a dominant part of the development effort for models. This is one of the reasons that your choice of a data platform has such a big impact. It's not just collecting and storing the right data, but also how you manage it, transform it, provide access, and maintain reliable versions that are critical for success.

Also keep in mind that even in fairly simple machine learning projects, it's not a matter of building "the" model and deploying it to production: data scientists normally work on many models at the same time, experimenting to find what works best in each particular situation. And, as we mentioned, the world will change, so ongoing monitoring along with a flexible system are important to be able to respond in a timely way as new models are needed. This need is why efficient management of multiple models is a fundamental requirement for successful AI and machine learning systems. Again, it's useful to have a data platform designed to handle these logistics at the platform level.

Most importantly, if the AI or machine learning system is to deliver value, you must plan on what actions (automated or human) will be taken based on the decisions made by the models. These actions are what connects the AI project to real business goals and real business value, as suggested in the previous figure.

As part of your preparation to plan for the resources, infrastructure, and team needed to implement an effective AI project, it is useful to familiarize yourself with some terms you are likely to hear, as covered in the following section.

Supervised, Unsupervised, and Semi-Supervised Machine Learning

These terms refer to whether or not training data is labeled with reference answers before training. Supervised learning uses labeled data that is generally applied in domains in which the data sets are well known. Classification is an example of supervised learning. The data scientist determines a desired output and trains models on data labeled with the correct value of that output. When deployed, the trained model is given new input data, and it produces an estimate of what the desired output would have been based on the data it saw during training. Supervised learning is a common technique to implement predictive modeling or predictive analytics, because it can learn the correlation between historical inputs and historical outcomes so that we can give it current inputs to predict future outcomes. Use cases include credit card fraud detection, spam detection, sentiment analysis, and medical diagnosis.

Unsupervised learning, which is also called descriptive analytics, finds patterns in unlabeled data. It may be less precise than supervised learning, but unsupervised learning can yield insights that might be impossible to discover otherwise. It may be used on complex or very large data sets that would be difficult for a human to analyze directly. Clustering is a machine learning technique that is an example of unsupervised learning. Algorithms such as k-means clustering find relationships in data; there is no single "right" answer. Different sets of clusters may be determined in the same data set depending on which related features are used along with their relative weighting. It's possible to use clustering to discover useful groups, and then assign those groups as categories in a new step using supervised

learning as classification. Unsupervised learning is often used in market segmentation, but can also be used to learn what normal operations look like for anomaly detection.

Deep Learning

Deep learning is a category of machine learning that has come to dominate certain kinds of tasks in recent years. In the most basic sense, deep learning refers to a class of AI and machine learning in which models learn in a cascade of layers (hence "deep"). For that reason it is sometimes called "hierarchical learning". The point of deep learning is to *learn* features rather than direct feature engineering. It's used in situations in which decisions would be too complicated to do effectively by more traditional machine learning methods. Instead, each layer in the deep learning model learns a transformation, using the output from the previous layer as input for the next layer of decisions. Deep learning is sometimes equated with neural networks, but neural networks are just one way to implement deep learning. Interestingly, the learning process at each layer may be supervised or unsupervised learning.

Deep learning is a very powerful approach that is being applied successfully to image analysis, speech recognition, machine translation, drug discovery, game playing, and many other problems with impressive results on previously intractable problems.

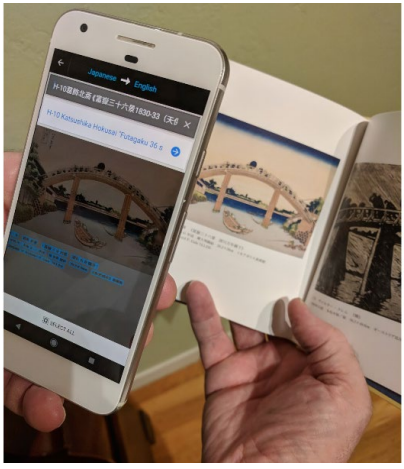


Figure 8. A machine translation app on a Google pixel phone is used to translate Japanese to English simply by pointing the phone at the text. This application is an example of image recognition and machine translation, both implemented using deep learning. (Image: © 2018 Ellen Friedman).

The development of deep learning models and tools originally required considerable sophistication on the part of the data scientists who developed them. The good news is that recently these approaches have become more broadly accessible in many areas such as image recognition. For example, the people at Google who developed the very popular tool called TensorFlow have done the heavy lifting for you. It's possible to get pre-built versions of TensorFlow image recognition models such as InceptionV3 that you can customize on your own images. To do so may require only basic data engineering skills, not sophisticated data science. In addition, it takes much fewer images to customize the pre-built model than were needed for the original training. This opens the way to benefit from these techniques in many situations across a wide range of industries without requiring a team of PhD-level data scientists to do it. This training of a pre-built image recognition model on new image data is taking advantage of a technique known as "transfer learning," which we describe in the next section.

Deep learning may be used as a component of a larger system. In reinforcement learning, for example, actions suggested by a model in a real or virtual world are recorded along with subsequent outcomes. These actions and outcomes are then used to produce a new model for suggesting actions. This is how the DeepMind program Alphazero learned to play both chess and Go at a better-than-human level. Few commercial systems have been based on reinforcement learning so far, but there appears to be amazing potential for solving difficult interactive problems.

Transfer Learning

The goal of transfer learning is to take advantage of what has already been learned in one task to make learning easier and better in a second, related task. Lisa Torrey and Jude Shavlik describe transfer learning as "...machine learning with an additional source of information apart from the standard training data: knowledge from one or more related tasks"[3].

The original task is referred to as the "source task," and it is used to transfer what has been learned along with new training data to train a related "target task." The model derived for the source task may be something pre-built, as in the case of InceptionV3 mentioned earlier, or it may be a model your team develops.

3 Torrey, L., & Shavlik, J. (2009). Transfer Learning. Handbook of Research on Machine Learning Applications and Trends. Retrieved from http://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf.

The models used for the source task and target task can be a variety of types, but generally the first task is used to simplify the input for the second task. Here's an example using synthetic data that shows how the results of a task improved with transfer learning with clustering model as the source task. The target task was a standard classification task with a simple model. The comparison shows receiver operating characteristics (ROC) which is a standard way to compare model performance. Permitting the final model to use the clustering model allowed it to perform considerably better.

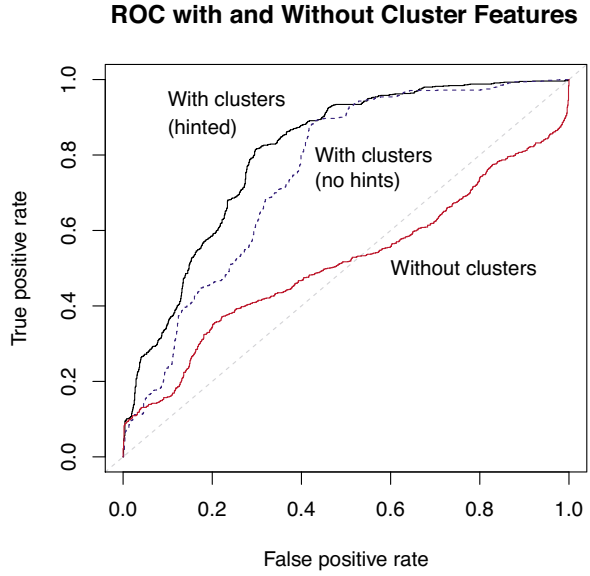


Figure 9. An example of improved model performance as a result of transfer learning. Source task was clustering; target task was classification. Both curves for "with clusters" showed higher ROC than the "without clusters" modeling, an indication of better model performance. Without transfer learning, performance was roughly comparable to random guessing. (Image: Ted Dunning).

We've mentioned that transfer learning often can be used for image recognition tasks. A different type of modeling might take advantage of transfer learning with a customer churn model as the source task and a target task of identifying customers who prove to be more profitable. There is a wide range of opportunities to use transfer learning, making it another catalyst for increasing adoption of AI and machine learning.

What Matters for Success with AI?

Once you've identified a practical business goal you want to address, the biggest challenge for success in AI is to handle the logistics of data and model management efficiently. This may not sound as exciting as experimenting with the latest algorithm, but it is the thing most likely to make or break the success of AI and machine learning systems. Moreover, much of that management should be handled at the platform level if you are going to succeed. For that reason, the choice of the data platform is important. A general tip to keep in mind is the data platform should provide support for everything in terms of flexibility in the choice of tools, data types and data access interfaces, data versioning, ability to run legacy applications, containerization of applications, and collaboration between team members.

In all of this, your highly scalable and reliable infrastructure, architecture and organizational culture should also offer you a strong degree of flexibility. AI and machine learning involve ongoing evolution, and you should be able to respond to changes as needed.

Data Scale, Data Access, Data Preparation

The quality of data for model training and as input for production models is a key factor in determining the performance and relevance of AI systems. Data can vary widely, ranging from unstructured text to highly structured streams, and come from sources such as IoT devices, text, log files, video, and audio. You need an infrastructure that can handle this wide variety, and being able to do so all in one system is a huge advantage. All this matters because data preparation and feature extraction are not only important, they also comprise a very large part of the effort involved in building AI and machine learning systems.

Data Preparation Is a Big Part of the Picture

Even before feature engineering begins to process raw data to become inputs for machine learning algorithms, a lot of work must have already taken place in the form of data preparation. Data preparation may involve exploration and clean up of data. Many of the big data tools such as Apache Spark, Apache Drill or Apache Hive commonly used for data preparation write output using the HDFS API, the native interface for the Hadoop file system.

This use of the HDFS API by tools such as Spark, Drill, or Hive might suggest to you that HDFS-based systems are ideal for machine learning applications, but in practice, the opposite is true. Most AI and machine learning programs do not read natively HDFS inputs. Instead, reading data using much more conventional POSIX interface for files is the most common approach. At best, HDFS support is very spotty. For instance, for R, there is a package that can read some columnar formats such as CSV files, but image files don't work and it is difficult to write many data formats to HDFS. This sort of uneven coverage applies to Python-based tools as well and can cause serious delays due to compatibility issues. It is usually better to stick to a platform that provides access to the same bits via multiple APIs, including POSIX file access.

Why is HDFS a challenge for machine learning libraries as well as many languages and legacy code? HDFS is in essence a write once, read only file system, and the HDFS API was designed to provide access to data across distributed Hadoop clusters. Hadoop is known for its scalability, but it has several important limitations, including issues with reliability and availability, poor support for small file sizes, lack of real-time processing, and incapability with legacy programs. There are other highly scalable, non-HDFS options for a data platform to support AI and ML systems, as we describe in the next section.

Once data preparation is done, feature extraction can get underway. Data will need to be available in a format accessible by a variety of tools such as Python, for which POSIX compliance is a considerable ease of use preference, for feature engineering. This is typically one of the most time-intensive aspects of machine learning model development. Feature engineering is critical to model performance because the choice of features directly impacts the quality of the result.

POSIX Is Standard for AI and Machine Learning Tools

POSIX (Portable Operating System Interface) is the standard for reading and writing files on Linux systems, and as such, it is the de facto standard used by AI and machine learning libraries, nearly all of which were originally created on Linux.

Unfortunately, Hadoop's HDFS isn't POSIX-compliant because originally it was only intended to support batch-oriented workloads in which files typically need to be written once and read many times. Supporting POSIX semantics, particularly the ability to update files, was outside the scope of the original, very limited mission for Hadoop. As such, HDFS is an append-only filesystem that doesn't support capabilities assumed by many programs.

Machine learning libraries often assume the presence of a POSIX filesystem. *That presents problems for HDFS-based platforms*. It doesn't completely prevent you from using HDFS, but data stored on it will be inaccessible to many machine learning systems. The most common workaround for this is to follow up on data preparation by copying data from HDFS onto a networked file system (such as a conventional NAS device) where it can be accessed by machine learning systems using a conventional file API such as POSIX. This workflow introduces many copies of data to make it available for machine learning. Having multiple copies of data on different clusters or devices can lead to confusion because it quickly becomes unclear which copy is the source of truth.

You can save a lot of grief in this respect if you look out for a data platform that natively supports POSIX file access as well as big data APIs including HDFS, as shown in the following figure. That way, you can do data preparation, feature extraction, and machine learning directly on the same data platform without having to constantly copy data out for processing and modeling and copy it back for storage. While you are at it, make sure the system meets your scaling and performance requirements as well.

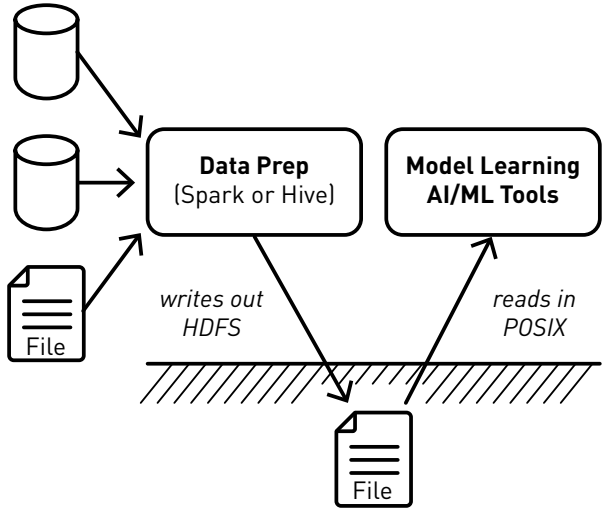


Figure 10: The best approach is to have a data platform that lets data preparation tools write output using the HDFS API and lets AI and machine learning tools read that output as input using the POSIX interface. Having these capabilities all in one system saves the burdensome extra step of having to copy data out of an HDFS platform into another system in order to for AI and machine learning models to use it. In Chapter 6, we describe a data platform with this capability.

Other Interfaces: Apache Kafka and S3

There are other ways to store data than just files that may be useful for data ingestion and long-term storage of large data sets. The open source Apache Kafka API provides message oriented streaming that is very useful for building streaming systems. This streaming API is supported by Apache Kafka, MapR Event Store for Apache Kafka, and Azure Event Hubs for Apache Kafka. Each of these are independent implementations of the same API. These alternatives also differ in deployment strategy. Kafka is typically deployed as a small, special-purpose cluster devoted to a few applications. Azure Event Hubs is only available as a multi-tenant managed service. In MapR, streams are built into the MapR Data Platform as first class objects embedded in the file system. These technologies are highly scalable, with excellent performance together with message persistence, but they differ in terms of how much throughput they offer, how well they handle high topic cardinality, and how long they can persist messages.

With Kafka or MapR Event Store for Apache Kafka, data from multiple producers, such as IoT sensors, is delivered to multiple consumer applications via subscription. Because data is not broadcast to consumers, it is available immediately for real-time processing, but it doesn't have to be used immediately. A consumer does not even have to be online when the message arrives; a consumer can be added later. This decoupling of data producers and consumers by these Kafka-style transport technologies means that a data stream makes an excellent lightweight connecter between microservices. A microservices architecture is not an absolute prerequisite for machine learning, but it does offer flexibility and makes deployment of machine learning based services easier. Although the Kafka API is not widely used directly by model programs and requires scaffolding to interface the model with a message stream, it is a valuable architectural option for AI and machine learning systems.

At the opposite end of the spectrum from the real-time oriented world of message streaming are object store APIs such as Amazon's S3. For large data sets that are not being frequently accessed, various options for cold storage or archiving can be helpful. S3, Amazon's flexible, scalable, object-based storage service is known for scalability and cost, but it typically must be used in conjunction with other forms of storage, particularly those that provide POSIX APIs. S3 is the dominant cloud storage service, but the API has been reimplemented by hardware vendors such as Hitachi Data Systems and as an alternative interface for files stored on MapR along with the HDFS and POSIX APIs. Unlike local or network-based storage, S3 is accessed via APIs, which requires special considerations in application development.

Raw Data and Data Versioning

While data preparation and feature engineering are essential stages in AI and machine learning development, it's also important to recognize that raw data is also valuable. You never know which features may be valuable for future projects or even for a current AI system. As data is refined to be appropriate for training models in one situation, you may be losing features that turn out to be key to good performance as the outputs of models undergo cycles of evaluation during development and after going into production. For this reason it's important to preserve raw data against being "corrected" or overwritten so that it can be revisited or shared with other projects. Data platforms that make this reasonably easy and cost effective are good choices for AI and machine learning systems.

In addition to recognizing the value of raw data, it's also important to have an efficient way to do data versioning, particularly for training data that you use to build models. As you do model-to-model comparisons, you need to know exactly what training data a model used as input just as you should record exactly what version of machine learning framework you used and the program steps taken to train the model. Version controlling the source code for your training model using something like git is easy to do, but version controlling your training data is much harder because it has a tendency to be much bigger (terabyte scale is not uncommon) than what is comfortable for version control software. Again your data platform could be your solution to this challenge if it is capable of making transactionally correct snapshots that are true point-in-time versions of data. With reliable snapshots, you can capture the training data just before starting the training process to guarantee reproducibility.

AI and Machine Learning Tools

Flexibility is paramount in AI development. There is no one tool or set of tools that excels at every task. Machine learning libraries are each optimized for different scenarios, so most organizations use more than one, and you will probably end up using several. In our experience, large organizations doing machine learning development use a minimum of five or more different frameworks, and we've seen as many as dozen in use in a single organization at the same time! That is not to say all frameworks are equally good, but there are many excellent choices, and they often excel in different situations. We discuss in more detail in this section frameworks that are good choices and are widely used.

Machine learning libraries Developers of machine learning applications have done each other a favor by putting their best efforts into libraries of code that can be assembled into powerful applications. Many of these libraries are available under an open source license, and companies such as Google, Amazon, and Facebook have contributed enormous efforts to communities surrounding different tools such as TensorFlow, MxNet, or Caffe. Most machine learning libraries have sweet spots for specific types of situations. Here are some of the most important ones that your machine learning platform should support. All are available under open source licenses unless otherwise noted.

TensorFlow Developed by Google, this Python library supports a wide variety of deep learning and neural networking models and algorithms. Its multidimensional array structure makes it an excellent toolset for pattern recognition applications, such as language translation and image recognition. TensorFlow is very scalable and can make use of GPUs if they are available.

H2O This extremely flexible library is meant to be used as an end-to-end solution for gathering data, building models, and delivering predictive analytics. It supports all popular machine learning development environments and data sources and allows for interaction with the data set during the training process. H2O can be used with Spark or R for additional flexibility and supports GPUs well.

R This open source scripting language for predictive analytics and data visualization is favored for its robustness, portability, and widespread availability under the GNU general public license. Developed in 1995, R has amassed a large and active user base, as well as large libraries of downloadable extensions. It continues to be a very popular choice among data scientists. R is generally not very scalable, although systems like H2O can be used with R to help with this. R is not normally used with GPUs, and most R libraries cannot use them.

Spark MLlib This is the machine learning library associated with Apache Spark. It provides simple learning algorithms and utilities for tasks such as classification, regression, clustering, collaborative filtering, dimensionality reduction, and model optimization. Version 1.2 of Spark introduced an enhanced and extended edition called Spark.ml that's intended to provide a uniform set of high-level APIs that developers can use to create and tune machine learning pipelines. That said, Spark's native capabilities for machine learning are well behind the capabilities of specialist systems like TensorFlow or H2O.

scikit-learn scikit-learn is a Python library that makes small-scale machine learning easier. scikit-learn is based on the widely used numpy and scipy which makes it easier to pick up for engineers already familiar with numerical programming in Python. Scaling is not a strong point for scikit-learn, however. Pytorch is often considered a relatively easy transition from scikit-learn to get better performance.

MXNet This deep learning framework is primarily used to train and deploy deep neural networks. It's known for its flexibility and scalability, and it is compatible with a wide variety of programming languages.

Amazon Machine Learning This paid managed service is as an easy way for developers to get started with machine learning without needing to have an extensive background in development. It provides guided data and analysis, model training, and evaluation, but is limited in scale and can't import or export models.

Microsoft Cognitive Toolkit Developed by Microsoft as a way to streamline its own speech and image recognition projects, this unified computational network framework can run on anything from a laptop to a multi-GPU cluster.

Torch and **Pytorch** Both are noted for their ease-of-use and flexibility, but not extreme scalability. Torch is primarily used for building scientific algorithms quickly and easily and is not limited to machine learning. Torch is unusual in that it is based on Lua instead of Python as Caffe or TensorFlow are. PyTorch is a reimplementation on top of the same C++ libraries as Torch, but using Python instead of Lua.

Caffe2 Developed by Facebook along with Torch and Pytorch, this library is optimized for deep learning on a variety of hardware platforms and is especially well-suited for applications related to image recognition and processing, with the ability to process over 60 million images per day on a single-GPU machine. Caffe is often used to productionize Torch/PyTorch programs.

New libraries are being announced all the time. It's important that the data platform you choose also can quickly accommodate these evolutions.

Collaboration and DataOps

Data science doesn't occur in a vacuum. The performance of the people who develop these systems is as important as the performance of the algorithms and models in determining what will be successful. Communication and collaboration among data scientists and others involved with AI and machine learning, such as data engineers, architects, business leaders and other stakeholders, are important for effective and agile development as well as for connecting machine

learning decisions to business goals. At a more technical level, model developers and data engineers may need to share details about data exploration and feature extraction methods, model training, and evaluation plans and visualizations of model performance. A data science notebook such as Apache Zeppelin or Jupyter can serve as a valuable aid to collaboration and communication although they aren't a substitute for good software engineering practices as models move into production.

Another powerful approach that provides flexibility and faster time-to-value is DataOps, an extension of the DevOps style of work to include more data-intensive roles such as data engineering and data science. It may sound like a soft skill, but experts who work with getting large-scale AI and machine learning systems say one of the biggest barriers to getting these systems into production is the lack of a DataOps approach. This is something worth taking seriously.

The advantages of DataOps include better focus on the end goal and more efficient use of workers' time and effort. The key to doing this is to change lines of communication to cut across traditional skill guilds. This means you often don't need to hire new people to fill out a DataOps team; you may just repurpose people with developer and data engineering skills and embed one or more data scientists as part of the team. When people with diverse skills see themselves as sharing a common goal, they tend to stay more focused on the goal and collaborate better.

Containers and Kubernetes

Few technologies have taken the enterprise IT world by storm as quickly as containers. You can think of them as miniature virtual machines that contain a basic set of resources needed to run an application. Though containers have been around for just a few years, their adoption is skyrocketing. In 2017, Gartner predicted that by 2020 more than 50% of global organizations will be running containerized applications in production, up from less than 20% today. If anything, that prediction was wildly pessimistic. For instance, a CNCF survey indicated that in 2018 nearly three quarters of enterprises already use containers in production.

What is motivating this rapid uptake of a relatively new technology? And what is the connection to AI and machine learning?

Containers were originally similar in intent to virtual machines, but, as Jim Scott pointed out in the publication *A Practical Guide to Microservices and Containers*, containers "can do a lot of what VMs cannot do." Containers are lightweight (or should be kept lightweight). Unlike VMs, containers can be launched or taken down very quickly and without a requirement for OS overhead. A container can be configured with all the supporting elements needed for a particular application, providing a customized, defined, and predictable environment in which the application will run. Equally important is that the environment provided for one application can be quite different than that configured for another application sharing the same resources, thus providing considerable isolation between applications. These qualities are much needed for truly multi-tenant systems.

In the context of AI and machine learning, containerization enables multiple models to be run simultaneously for comparison purposes or to have new models ready to deploy. Containerization of model applications also reduces the differences between testing and production environments, making model evaluation more accurate. Additional flexibility is provided by containers because they are portable, providing developers with an easier way to deploy and transfer applications. For example, with containerization, an application developed on premises can be easily moved to the cloud and vice versa.

The Really Big Story

Surprisingly, the containerization of conventional applications is not the really big story. Instead, containers have paved the way to build what are called cloud native systems in which services are implemented using small clouds of containers. Ideally, each of these services can be independently deployed and upgraded in a rolling fashion without ever taking down the service.

Doing this across shared resources, however, requires effective management and orchestration. With rapidly increasing adoption, Kubernetes is emerging as an industry leader for this purpose. Kubernetes was originally developed by Google and now is available as an open source orchestration layer for containerized applications, automating deployment, scaling, and operational monitoring and management.

Containers are ephemeral, however, so the lifecycle of data is longer than the life cycle of individual containers and usually longer even than containerized services. It is a core assumption of implementing services in containers that containers can be started and stopped easily and quickly. This assumption leads to one of the key challenges of working with containers: they work best only if they are stateless. Containers need to be lightweight in order to be quickly deployed and portable. How, then, can you avoid being limited to stateless applications when taking advantage of containers? The trick is to persist data (state) outside of any application container to a data platform. Keep in mind that the output data for one containerized application may be the input for another, as depicted in Figure 11.

In fact, it is important for containers to be able to use any of the dominant forms of persistence. They should be able to use files, tables, and streams as appropriate without adding bespoke clusters for each additional use. Indeed, files, tables, and streams should be persisted *to the same shared platform*.

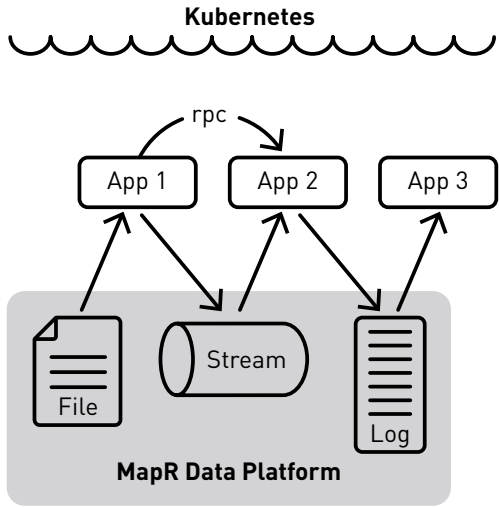


Figure 11. Kubernetes is a leading orchestration layer for managing containerized applications. To take full advantage of containers, a data platform that can persist state from these applications is needed. The MapR Data Platform, depicted here, is a unique example of a platform that can persist files, tables, and streams in the same technology and works in coordination with Kubernetes to orchestrate persistence of state from containerized applications.

This data platform needs to be able to do more than simply store data: it needs to orchestrate data. Kubernetes handles the orchestration of applications, positioning containers to maximize efficiency and naming services, while the data platform is like Kubernetes but for data. The data platform serves as the data orchestration layer to complement what Kubernetes is doing for computation.

Capabilities That Matter Most in the Data Platform

We may not know what big innovation is coming down the pike, but we can prepare for it anyway. A good AI data platform should have a track record of success in handling current and future technologies for AI and machine learning as well as for data preparation and feature extraction *without the need for filters, workarounds, or copying data out to other machine learning or analytics platforms*. Remember, many machine learning libraries don't work natively with HDFS. This limitation can create the need to use multiple clusters, an unnecessary complication and expense. In contrast, the importance of direct workflow from data ingestion and data preparation to the direct access by AI and machine learning tools — and the data platform's role in making this possible - was shown previously, in Fig 11. True POSIX compliance in your data platform is a key advantage for AI and machine learning as well as for the whole range of other analytics applications you plan to run.

A platform capability almost as important as universal data access is to *go beyond data storage to provide data orchestration*. This term refers to a range of key data actions, many of which we have described throughout this book, that should be pushed down to the platform level rather than having to be programmed at the application level. Data location and movement, core security, accurate point-in-time views of data such as snapshots, mirroring, tiered storage based on usage frequency, and support for real multi-tenancy including data persistence for containerized applications are all aspects of data orchestration.

A data platform suitable for AI and analytics should separate the concerns of data scientists and developers from those of IT administrators. In all this, the platform of choice should provide simplicity, both for management and in the architectures it supports.

Auditing and Security

In modern organizations, it isn't enough to process data. You have to be able to document what data went into any material decisions whether they were automated or human. You also have to document how you processed that data and show you have sufficient control over your data to avoid privacy leaks or data corruption. Essentially, you have to track and protect all the data your organization imports or generates and figure out how it affected the operation of the business.

To do this, you need audit tracking that records operations on persisted data. This tracking needs to provide you with the raw information about what happened and which program did it. This information should be more granular than just "opened a file," by giving information about whether any data was actually read or written. It is vital that these audit logs be comprehensive and consistent across all the kinds of data persistence that you have, including files, streams, and tables. In fact, fracturing audit information across lots of small clusters is one of the strongest reasons to avoid cluster sprawl.

In addition to recording what actually does happen, you also need to be able to tightly control what can happen. This is done by setting permissions on directories, files, tables, and streams (and volumes if you have them). It is important to have effective control over data access and update at a very low level rather than depending on faith that an application-level process can prevent programs from intentionally or accidentally saving bad data or allowing access to data that shouldn't be visible.

There are two general styles of access control in distributed data systems. The first, perimeter security, depends largely on gateways that carefully inspect each query of your data for possible security violations. In the Hadoop ecosystem, perimeter security is implemented by open source components such as Apache Ranger or Apache Sentry.

The second major style of access control is known as core security. Core security secures the data elements such as files, tables, and streams themselves and controls access based on the ID of the user running a program and the permissions on the data itself. We recommend that you look for a data platform that can express permissions in terms of users, groups, roles, and policies so that you have ultimate expressiveness and can manage large-scale systems easily.

Perimeter security depends on all access going through a single choke point and can only control access via languages it understands. Typically, this means access to data must be via SQL queries. This is untenable for machine learning applications since very little machine learning is done using SQL to access data. SQL may be used in creating training data sets, but for the machine learning itself, it is almost certain that data will be read directly from files. This completely circumvents perimeter security. Another problem is that perimeter security often requires data be accessed using a gateway process, which can be a severe bottleneck in terms of how fast data can be extracted.

If your data platform doesn't support multiple access methods for data, you could have a problem with security as well. This arises because one of the most common workarounds for, say, the problems in accessing HDFS data using normal file I/O is to copy data to a different data system. The issue is that now you have to manage permissions on two systems, and the methods for doing this probably are different, possibly even different in how you can express permissions. This also arises when using message streaming with Apache Kafka since the permissions in Kafka are very different from the permission schemes used with files. It is much easier if you can simply avoid the issue by having all of your data in a single platform.

In summary, you need to have a unified audit system that allows you to get access to a history of which programs did what to which data systems. This audit trails has to record events for all of your data, whether in streams, tables, or files. As well, your data platform should allow unified access control for all the kinds of data you want to work with including files, tables, and streams by using standard concepts such as application user ID, group ID, access control lists or expressions, roles, and policies. Perimeter security is insufficient for access control with machine learning. By implication, you also need to avoid cluster sprawl as much as possible because that will massively complicate both the auditing and access control problems.

Separation of Concerns

Complexity in the form of having to track where data is located should be beyond the scope of data scientists and model developers. Projects move more quickly and with less errors when data scientists can focus on the process of planning, developing, training, and evaluating models

and interpreting their outputs rather than having to deal with a lot of housekeeping that is programmed application by application.

Instead, an AI data platform should abstract this complexity away from data scientists and IT administrators. It should provide a common set of data services for high availability, disaster recovery, security, and multi-tenancy with a single file system that accommodates structured, unstructured, and semi-structured formats as well as streaming data. It should further provide a common set of APIs that developers can use to access data without worrying about the underlying source.

This ability to separate the administrative concerns of where data is, how it is replicated, and how it is stored from the functional concerns of what you actually want to do with the data is the essence of what we mean when we say that you need a data platform to orchestrate your data.

Checklist Guide for AI and Analytics Data Platform

This section is a very terse checklist of the capabilities we think are important for you to have in a data platform for AI and analytics applications. All the items here have been covered in a more expanded form in the rest of the book, but having these items in one place can be helpful when you are reviewing options.

Before we start the checklist, there are two quick things to note. First, legacy systems aren't going to cut it when you sit down to build your system largely because they lack a data platform suitable for the modern needs of AI and machine learning. Second, we make specific recommendations in Chapter 5 about how you can meet the requirements in this checklist. Following those recommendations may be easier than tracking through a long checklist and iterating your architecture.

In general, the market is full of point solutions, each of which may be excellent in its own right but may not work all that well together. The data platform is the foundation for all the tools you will be using, including cloud platforms. Building on a solid foundation that is flexible, scalable, and adaptable to change is the most cost-effective decision for the long term. The ability to deploy seamlessly in a hybrid cloud environment while also adapting to the power of new technologies like containers and serverless computing will minimize future integration roadblocks.

Here are some things a data platform should be able to do:

- ✓ Support diverse computational tools and frameworks such as Apache Spark and Apache Drill for data preparation as well as a wide (and expanding) range of machine learning libraries. In practice, this means your system should provide access to the same files using both traditional file APIs (i.e., POSIX) as well as HDFS APIs without sacrificing performance.
- ✓ Be able to run AI and analytics applications simultaneously on the same data stores without requiring multiple clusters, silos, or storage appliances. This translates into faster time to market, reduced maintenance, smaller support teams, and faster response. This capability requires POSIX compliance as well as HDFS APIs.
- ✓ Support a wide variety of data types and sizes, including both large and small files and structured and unstructured data delivered in tables, streams, or files. The platform should also support a wide range of ingest mechanisms, including NFS, cloud object storage, Apache Kafka compatible streams, and direct access via POSIX or other HDFS APIs.
- ✓ Scale to handle exabytes of data and trillions of files, both large and small, in a single cluster with full read-write semantics and automated placement and movement of data.
- ✓ Allow you to build a comprehensive data system to integrate data across on-premises computing, any or all public clouds as desired as well as any edge clusters you need.
- ✓ Provide integrated, platform-level security and auditing to help you to comply with regulatory standards like PCI, HIPAA, NIST 800-53, GDPR, and ISO 27001. Such controls should be integrated, not bolted on, and should address permissions at the file, directory, volume, column, document, and element level by user, group, policy, and/or role. Support optional integration with popular authentication and directory systems such as Kerberos and LDAP.
- ✓ Integrate tightly with Kubernetes with support for conventional, legacy applications in the same cluster with big data systems and machine learning systems.
- ✓ Support high levels of data multi-tenancy so multiple high-performance applications can run simultaneously, share data across application types, and leverage previous data preparation efforts.

- ✓ Support the very high performance levels required by GPU software. You will also need to have data placement control to allow you to make effective use of GPUs without splitting your data across different kinds of clusters.
- ✓ Provide integrated NoSQL database that natively supports JSON document and wide column data models with high performance, low latency, strong consistency, multi-master replication, granular security, and automatic self-tuning.
- ✓ Provide integrated message streaming support. Compatibility with the Apache Kafka API is important here so emerging trends like streaming microservices can be implemented cleanly.
- ✓ Be compatible with all popular public infrastructure-as-a-service platforms, including Amazon, Google, and Microsoft.

Benefits of a Good AI & Analytics Data Platform

Choosing a data platform that adheres to the checklist above yields the following benefits:

Lower Total Cost of Ownership (TCO) and Platform Cost

The TCO of a software system with significant machine learning or AI components can be significantly different from normal software system costs in a number of ways, but it still breaks down into the same general categories. The major divide is between development and deployment. Within each category, there are further breakdowns into platform costs, team costs, IT impacts, and indirect impacts. The IT impacts come from increased scope of support for the IT team. The indirect impacts begin when your data teams find previously unknown problems in data as they examine it more closely and use that data in new ways. Indirect impacts also arise when the data teams start stressing source systems in the process of extracting data.

One major difference is that for normal software, development is generally done on hardware that is considerably smaller and cheaper (sometimes just a laptop) than the system used to deploy the production software. With machine learning, in contrast, it is common for the system involved in model training and data preparation to require much more computational power than the final system that uses the models. This is particularly true when models require GPU power to train, but

only need CPUs to use. It is important to keep this inversion of costs between development and deployment in mind precisely because the small development hardware/big deployment hardware assumption is so strongly ingrained.

If your organization strictly limits available machine configuration, available configurations may be a poor match to the machine learning computational loads. This can inflate platform costs substantially. For training machines, this shows up as machines with insufficient memory, not enough computational cores, or poor memory bandwidth. For machines hosting your data platform, this commonly manifests as difficulty getting machines equipped with enough storage devices or with high enough performance storage devices.

On the other hand, the high computational loads involved in building machine learning models are often very intermittent. As a result, This can mean that if you have a platform that supports multi-tenancy well, sharing hardware can be very cost effective. In general, this should be a key cost-reduction strategy. Good multi-tenancy can result in a number of virtuous cycles as well. For instance, better sharing usually means you have fewer copies of data, which makes governance easier.

Whether to use cloud should be largely a cost decision, but it may be strongly impacted by organizational policies (either pro- or contra-cloud). Security considerations may preclude any use of public cloud. Deployment policies might, in contrast, require the use of public cloud. If you are developing large models on-premises, it is likely that using a system like Kubernetes to build a private cloud will be useful. Regardless of policy, budget may override other considerations. We have seen organizations with strict cloud-only policies retreat to a cloud-almost-always policy in order to save costs on GPU hardware.

Your mileage will vary.

Support for Model Building Strike Teams for DevOps

One key component of the TCO for machine learning systems is that you will need some expertise in building machine learning systems. The level of expertise needed is declining quickly as machine learning becomes more democratized, but you still need someone who can help teach the rest of your team how models are built, deployed, and evaluated and how to prepare the data needed to feed the process. For the medium

term, this experience is rare enough that although you may not require somebody with a PhD, you will need somebody. Many organizations require outside help until they can develop their teams' skills internally. One particularly effective strategy is to temporarily embed a data scientist in a team with business domain expertise. As the team becomes comfortable with data preparation, model building, and monitoring, the data scientist can move to a new embedded position. Having a separate data science team is typically much less effective unless the organization has very substantial historical data science capabilities.

In any case, make sure you support your model building teams with others who have or are developing data engineering skills. Surveys show a very large fraction of data scientists' time is spent on relatively mundane data prep tasks that could easily be shared across a wider team. A good data platform can facilitate this sharing and make it easier for teams to collaborate on the various tasks required to train and deploy models.

Reduce IT Support Burden

From the IT point of view, there is only one rational strategy and that is to push hard for multi-tenancy on a very high reliability and high availability system. No other strategy allows the IT organization to keep their personnel costs flat as system usage increases and the system needs to be scaled up (as it very likely will be). If done well, a very small team can manage a very large shared resource. Among MapR customers, we have seen teams as small as three support clusters with thousands of machines used by hundreds of analysts and developers.

In contrast, if systems are used that do not support very high levels of reliability and multitenancy, you can expect IT support burden for machine learning systems to scale with the number of applications. Systems based on HDFS can be particular sore points in this regard because of their limits on file count and because reliability problems and support loads increase radically at critical scale levels. That means budget factors derived when systems are new can turn out radically wrong as systems go into production.

Making the Business Case for AI

An old joke says the way to eat an elephant is one bite at a time. The same applies to getting any big new initiative past skeptical executives, and AI projects are no exception. It's a good idea to break the issue down into bite-sized pieces.

As you formulate a way to make the business case for an AI or machine learning project, keep in mind that it is unusual for these types of systems to be the main focus of a business. That doesn't mean they won't deliver value. As we have described, done right AI can save millions by relieving bottlenecks in business processes and reducing costs through automation. AI and machine learning also may open new lines of revenue. But these approaches usually are adjustments or additions to primary business goals. For that reason, it's important to have an overall architecture and infrastructure (including the data orchestration layer) that can support AI and machine learning initiatives on the same system as core business function. If you don't have to build a new system to support AI and machine learning, these potentially high value but speculative projects are much more likely to be feasible in terms of direct costs and IT impact.

Avoid talking about excessively grand visions at the outset. Look instead for small projects that can show immediate results in a way that's understandable to non-technical executives. A good approach is to start with one of the many commercial AI-based services. For example, X.ai can automatically negotiate the scheduling of meetings by handling interactions between participants. If your organization uses Salesforce.com, its Einstein assistant can recommend customer contact options based on previous experience. Passage.ai has a toolkit

for developing intelligent chat bots. Many examples are available as video demonstrations or offer free trials that are sufficient for building a basic application.

All of the commercial cloud vendors also offer free trials to enable you to build simple models using your own data. You can take advantage of their extensive machine learning libraries to develop simple proofs of concept based upon your own data at little or no cost.

Cloud platforms are a good way to get your organization started with AI development. Costs are modest and there is an ample supply of coursework, libraries, and code samples. Working strictly in the cloud can be limiting, however, particularly when building models based on production data. As noted earlier, copying data to and from target platforms is a practice to be avoided. Building models on a container foundation in the cloud gives organizations the freedom to move projects behind the firewall, as well as between commercial cloud services. Hosting sensitive training data and the models based on that data in the cloud may be problematic for your security team as well.

One of the most important things to keep in mind is that the value returned from an AI project does not scale linearly with the complexity of the project. A very simple model can be worth as much or more than a complex one if it's the right model for the situation. The ability to recognize where in the business process an AI or machine learning step could be beneficial is as important a skill as the ability to actually build the model. When building proofs of concept, look for pain points in the organization. For example, organizations that struggle to classify a lot of images could see huge labor-saving potential in automated image tagging. Those with large outbound email operations will see value in automated message customization. Companies with contact centers will see the payoff of software that classifies and routes questions based upon linguistic analysis. As tempting as it is to show off gee-whiz technologies, focus instead on those that have clear bottom-line value.

When launching your first internal development efforts, be aware of the need for collaboration and transparency. Data has many masters, and in this era of self-service, end-users are more empowered than ever to lay claim to the data they need and how it's used. Recently, a discipline called DataOps has emerged that applies the disciplines of DevOps to data-based projects. Predicated upon principles of cooperation, reuse,

quality management, and continuous improvement, DataOps recognize that all stakeholders need to have a voice in how analytics is developed and applied to the organization's data, and that there is no room for opacity or black-box solutions.

Building from the foundation of small projects with quick turnarounds, you should find yourself gradually taking on more ambitious tasks. The trick is to keep the momentum going and not let the AI initiative sputter after a year or two, as often happens when the supply of easy projects has been exhausted. But most importantly, *choose projects that fit real business goals, whether they are easy or difficult*.

MIT Researcher George Westerman has written about six factors that contribute to the success of innovation teams: sponsored, separate, small, systematic, shared, and seen. These are good practices to adopt in your fledgling AI efforts. Executive sponsors help nurture small teams through the sometimes rocky startup process, introduce them to influential stakeholders, and protect them from detractors. A systematic approach ensures that potential projects are given fair consideration and decisions are made quickly. Small organizations are less likely to endure budget-cutting pressure than large ones. Collaborating with business stakeholders improves buy-in and creates shared commitment.

A little self-promotion can help. Business users will be intrigued by what you're doing, so communicate often about what AI means to your organization and the results you are seeing. Be aware of the possibility of apprehension about AI as a threat. When discussing your successes, focus on those that cut down on drudgery and make work more fulfilling. Show how AI can grow the business rather than cut headcount.

MapR: Who We Are and How We Can Help

What aspects of AI can be made easier by your choice of technologies? This ebook has highlighted a number of ways in which the capabilities of the data platform you select can have an impact on the success of your AI and machine learning projects. This chapter explores in practical terms how a specific example, the MapR Data Platform, meets these challenges by providing **dataware**. We'll explain what that means and what the implications are for AI and machine learning.

MapR Basics

In order to assess what MapR offers as a modern platform for large-scale AI, machine learning, and analytics systems, it's important to understand a few basic characteristics of MapR and the ways in which it differs from other big data technologies.

Put simply, *MapR is dataware*. It goes beyond just scalable storage to provide a data orchestration layer. This results in the flexibility, speed, scale, reliability, and convenience of administration needed to support successful development and production AI systems across locations from multi-cloud, hybrid, and on-premises data centers to computing at the IoT edge. Three of the most important characteristics that distinguish the MapR data platform as a good foundation for AI and machine learning applications are these:

- All major AI and machine learning systems can directly access and model data from the MapR platform, letting you use a wide variety of AI/ML tools.

- With MapR, you don't have to copy data out of the data platform to a data science platform for processing as you would with an HDFS-based system.
- The MapR Data Platform is unique in being be able to persist data from containerized applications as files, tables, or streams all in one system, making it easy to leverage the flexibility and convenience of containerization from edge computing to on-premises to multi-cloud deployments.

These are not the only ways in which MapR dataware is suited for AI applications, and this chapter also explores a range of other ways MapR can be used for these systems, however these capabilities are fundamental to building performant AI systems. For this reason, let's start with them and see why they matter and what about MapR makes it able to provide these three unusual advantages.

The MapR Data Platform is an enterprise-grade technology that provides distributed, highly scalable data storage with built-in capabilities for files, tables, and streams accessible via standard APIs including but not limited to popular open source APIs. One point here is easily misunderstood, because many other systems use the term data platform. Keep this in mind: with MapR, "built-in" means that files, tables, and streams are *literally all part of the same technology, the same code, able to coexist on the same cluster*. These are not separate components intended to work together through connectors, as is true for other systems. Think about the implications of this. If you want to work with streaming data, for example, with MapR you do not need a separate cluster for stream transport from the main cluster where you carry out data processing and modeling. MapR Event Store for Apache Kafka, the stream transport feature that supports the open source Apache Kafka API, functions directly on the same cluster as the stream processing and modeling or analytics applications. This difference eliminates the need for a separate cluster for stream transport, which is a huge simplification. Similarly, NoSQL databases and files coexist on the same cluster as well. Simplification in architecture and infrastructure is a big advantage, reducing overhead in hardware and administration.

The key role, then, for MapR in AI and analytics is to serve as a foundational platform that provides a comprehensive view of data with unusual ease of access, whether on-premises or in cloud or multi-cloud deployments.

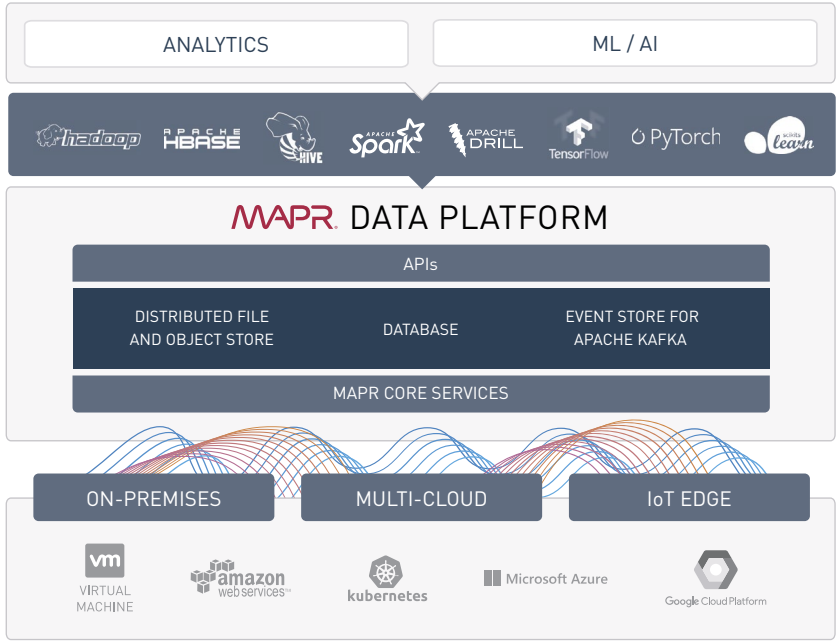


Figure 12. The MapR Data Platform combines a fully read/write distributed file system with the unusual features of a built-in NoSQL database and built-in stream transport. Data handled by MapR is directly accessible by AI and analytics tools, legacy programs, and via modern open source APIs. The MapR platform serves as dataware useful for building a comprehensive data system across on-premises, multi-cloud, or hybrid data centers.

How does the MapR dataware make it possible for you to use your choice of AI and machine learning tools to directly access and model data at scale without having to copy it out to another system? The answer lies with the nature of the MapR file system, which combines reliable and standard data APIs with a highly scalable distributed system.

Standard File Access Provides Flexibility Through Ease of Data Access

The point of a modern distributed system is to make it possible to scale for performance and size, and to do this in a practical, cost-effective way. To be practical in a business sense, the scalable distributed system also needs to be reliable and highly available. And functionally, it's also important to be able to have easy access data by a variety of tools and applications, including those used for AI and machine learning systems, without a large overhead in management costs or having to copy data from one system to another.

MapR was designed to meet these requirements with a file system (MapR XD Distributed File and Object Store) that *provides full file read and update capabilities using standard APIs*. MapR XD uses POSIX APIs to access data, and the result is convenient and normal file access even with a distributed system. For example, MapR XD allows all code that runs on Linux to access data stored on a MapR cluster. Essentially all code being written today for machine learning and AI applications can access data directly in the MapR Data Platform. This is in sharp contrast to what happens with AI/ML trying to access data from HDFS-based platforms. In addition, MapR allows open access via standard and open APIs. You can, of course, access data using Hadoop APIs, but you are not limited to Hadoop access. That opens the way to full freedom in your choice of AI and machine learning tools as well as standard analytics technologies. Developers can use Python, R, MLlib, H20, TensorFlow, and more to build applications with direct access to data from the MapR platform, and they can mix and match systems that use standard (POSIX) file APIs as well as Hadoop (HDFS) APIs.

This flexibility for current and future AI tools does more than just keep developers and data scientists happy (although that's a useful advantage). MapR's POSIX compliance and full file read and update ability for multiple data structures - files, tables, streams - in the same system keep overhead in data and application management much lower. This is a particular advantage in AI and machine learning systems that need a comprehensive view of large amounts of data often from a variety of data sources and that require management of many models at the same time.

In contrast, trying to run AI projects on HDFS-based systems creates large amounts of overhead just to try to get to data by whatever access is normal for the AI/ML tools of choice. It may surprise you that many other data science systems require copying data to a non-primary cluster for processing and modeling and then copying results back to the primary cluster. This not only slows down data scientists, it also creates additional unwanted overhead in just managing which copy of your data is the authoritative one. The simplicity of the MapR's standard file access avoids this problem altogether.

MapR's Distributed Metadata for Reliability and Performance

MapR is not only unusual in offering POSIX compliance for standard data access in a distributed system, it's also very unusual in providing fully distributed metadata. Why does this matter? It has to do with reliability, availability, and performance at scale. MapR was engineered to have the advantages of scalability at low cost but without the limitations of a distributed system such as the write-once/read-only HDFS system. Not only does HDFS not allow data access using standard APIs, thus forcing you to move data for processing and modeling, it also has problems caused by centralized handling of metadata in a NameNode process rather than using a distributed design. The potential single-point-of-failure problem with an HDFS NameNode is slightly alleviated by using a redundant NameNodes, but with HDFS you still risk losing a large amount of data. Equally worrisome is that centralizing metadata also centralizes metadata updates, leading to a network traffic jam. MapR avoids the NameNode problem altogether by not having a NameNode at all and instead supporting the distribution of metadata across the cluster.

Another way in which the basic capabilities of the MapR Data Platform address the requirements of AI systems is by making it possible and practical to persist large amounts of data, including event-by-event streaming data, long term. This capability is particularly important for AI and machine learning because you don't always know which data features will be important for new models or future projects. For event-by-event data, the fact that MapR's stream transport is built in means message data is distributed across the cluster. That, in turn, makes it practical and cost-effective to set the time-to-live for messages to years or infinity, even at large scale. Doing so gives you a long-term, replayable event-by-event history, a resource that can be extremely valuable not only for compliance but also as input data for AI systems.

Ease of Data and Model Management

One key to getting real business value from AI and machine learning systems is to have efficient data and model management. MapR provides many of the capabilities needed for the platform to take on much of the burden of logistics for these systems, freeing data scientists to focus on the modeling and insights, which are their forte. This is another aspect of dataware being appropriate for AI systems. MapR provides a powerful data management feature known as a volume. The MapR volume is essentially a directory with super powers, and this has big payoffs for AI and machine learning.

MapR Volumes for Controlling Multi-Tenancy, Mirroring, and Data Locality

The MapR volume houses directories, files, database tables, and streams together, all in the same management abstraction. Policies such as permissions for data access for users and groups can be conveniently set at the volume level. This makes it much more practical to run a multi-tenant system, and that's a big advantage. Different data science teams can be working with separate applications but still share the same cluster without interfering with one another.

In addition, the scheduling of incremental mirroring is also volume-based in MapR, making it easy to have multiple data centers for disaster recovery planning or to deploy between on-premises and cloud or multi-cloud clusters. MapR administrators set up many volumes on a single cluster, making fine-grained data management convenient.

The MapR volume is by default distributed across the cluster, but this is configurable. You can use the volume as an easy way to establish data locality, such as placing certain data on special hardware including the GPUs that are useful for deep learning and other AI systems.

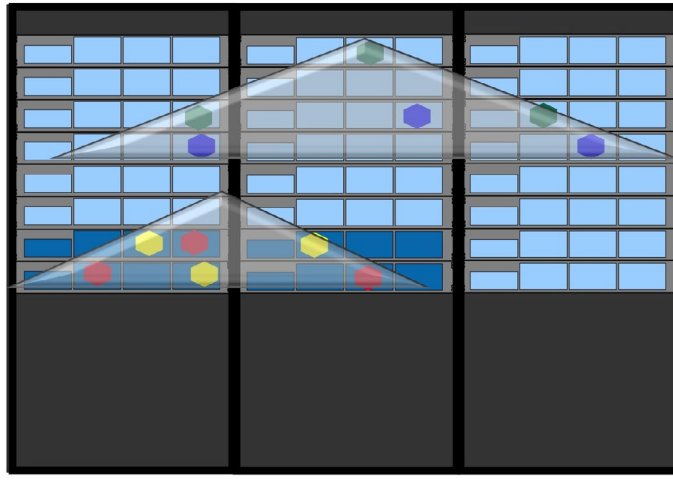


Figure 13. Data management for the MapR Data Platform is made convenient through an abstraction known as a MapR volume, depicted here as a transparent triangle. Volumes contain multiple types of data (files, database tables, streams) and provide an easy way to set policies such as data access, mirroring, snapshots, or data locality. Here, one volume is restricting data placement to specialized hardware (shaded darker in the figure).

MapR Snapshots for Data Version Control

The ability to have a way to keep accurate point-in-time versions of data is particularly important in AI and machine learning systems, although this key capability is also useful for analytics in general. For example, data scientists need to be able to easily and accurately find the exact data seen by a model or outputs for a specific iteration of the system or particular condition of the environment. MapR snapshots, in contrast to some systems, are consistent, non-leaky snapshots, providing an accurate point-in-time version of data stored in a volume. Snapshots are easy to set up manually or on a schedule, making them ideal for imposing version control on your data, giving you a repeatable process for building models. The MapR volume is also the basis for making snapshots.

Global Namespace

Another basic but unusual feature of the MapR Data Platform is its global namespace for all data structures. The global namespace makes it easy to view multi-site clusters as one logical cluster, a big convenience for developers, data scientists, and administrators who use MapR. For AI, this is another way that MapR makes it easier to get access to a comprehensive view of data, avoiding a patchwork of data silos. The potential to see a full view of data can make a big difference for all types of applications, but it is especially useful for AI and machine learning systems that often require large datasets from multiple sources for accurate modeling. The global namespace supports the separation of concerns mentioned in the introduction to this chapter, and it is a key capability for building a global data system, as discussed later.

Streams Support Microservices and Multitenancy

How do you take advantage of the efficiency and cost savings of a truly multi-tenant system? One way is through the independence offered by a microservices style architecture. Apache Kafka and MapR Event Store for Apache Kafka (which supports the Kafka API) deliver high performance at large scale with message persistence - the key combination needed for stream transport to serve as the lightweight connector between microservices. Kafka runs well on the MapR Data Platform, or you can have the additional advantages of the built-in MapR stream transport. Either way, it can be useful for AI and machine learning systems both in development and in production to use a stream-based microservices architecture.

This streaming microservices approach is also a good basis for data and model management frameworks that efficiently handle machine learning logistics. This design provides the flexibility and agility needed for continuous and speculative model deployment coupled with ease of model rollback as necessary. Multiple models can share data from the same stream without interference. MapR streams also provides the long-term replayable event-by-event history that can be valuable for AI systems. The Rendezvous Architecture is just one example of this type of data and model management framework, and the MapR Data Platform is very well designed to support management approaches such as this.

MapR streaming also makes it easy to deploy models at the IoT edge as needed and to convey data aggregates or partially processed data back to data centers on-premises or in cloud. This design can be a powerful advantage where highly distributed data sources are involved or action needs to be taken locally at many sites.

MapR + Kubernetes: Powerful Pair for Containerized Applications

Another key aspect of flexibility, multi-tenancy, and convenience in deploying models is the use of containerized applications. Containers provide a way to run applications where you want, each in its own predictable, repeatable environment. Multiple containerized model applications can be run at the same time, in different settings, without interference. Containerized applications also make it easier and more accurate to do model-to-model evaluations.

As mentioned previously, Kubernetes is emerging as a leader in the orchestration of containerized applications, and as such, it is a key technology for AI and machine learning systems. But to be effective, it's important not to limit containerized applications to those that are stateless, that is, those that do not need to persist data. A much more useful system is one in which the applications orchestrated by Kubernetes can access multiple data structures from a data platform and persist output to a data platform.

MapR addresses this critical need by being able to do just that. Containerized applications can access or persist data as files, tables,or streams, all using the same data platform, as shown in Figure 14.

The combination of Kubernetes and the MapR Data Platform form a powerful pair for taking advantage of application deployment via containers, either on-premises, in cloud and multi-cloud, or as a hybrid on-premises/cloud architecture. Kubernetes provides the orchestration layer for containerized applications, and MapR acts as the dataware needed for data orchestration. In this context you can think of *MapR as being like Kubernetes for data*.

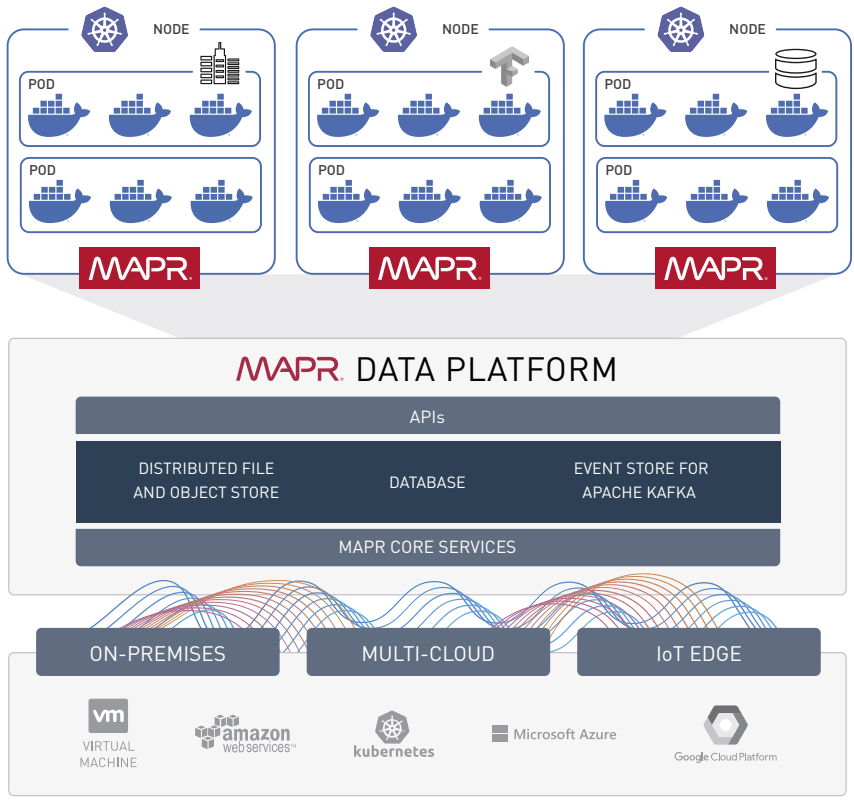


Figure 14. Containerization of applications is useful for ease of deployment, for multi-tenancy, and to have a predictable, customized environment in which each application runs. State from these applications can be persisted as files, tables, or streams in a single system: the MapR Data Platform.

MapR Dataware Seamlessly Spans Edge to On-Premises to Multi-Cloud

The need for a complete view of data without unwanted silos, combined with convenient access even from remote locations makes it desirable to build a global data system that spans clusters from cloud and multi-cloud deployments to on-premises data centers and even to remote data sources such at near sensors at the IoT edge. This approach can make both development and production more efficient and more cost effective. With a global data system, not only are current applications more likely to be successful, but it's also easier to take advantage of new opportunities. MapR is dataware with the capabilities needed to span edge to on-premises to cloud and multi-cloud for deployments in all these locations. The comprehensive nature of this system means you can build new applications or embark on new lines of business without having to build a whole new system. This characteristic is particularly attractive for speculative but potentially high-value projects, which is often what AI and machine learning represent. *By reducing the entrance cost for AI and machine learning, you reduce the risk of new endeavours and make it more likely for them to return value*.

Data Science Refinery: Better Collaboration Among Data Scientists

Building AI and machine learning applications on the MapR Data Platform provides extreme flexibility through direct access to data using a wide choice of AI/ ML tools, and MapR also offers a pre-configured Data Science Refinery (DSR). DSR is a suite of popular open-source data science tools in a pre-configured offering that can easily be distributed to many data science teams in a multi-tenant environment. Using DSR can improve productivity of data science teams through:

- Convenience
- Collaboration
- Agility and flexibility
- Out-of-the box secured access to all data

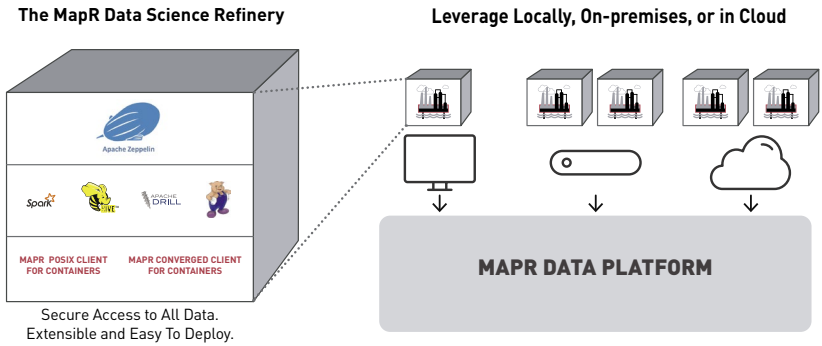


Figure 15. Example of the tools that can be pre-configured in the MapR Data Science Refinery: Apache Zeppelin notebook is an excellent way for data scientists to collaborate and the collection of other data science tools can have secure access to data assets of the MapR platform, across locations from edge to on-premises data center to cloud deployments.

Convenience starts with DSR being a preconfigured Docker container that provides easy deployment. DSR leverages the MapR platform as a persistent data store so you can run stateful as well as stateless containerized applications. It also makes it easy for data scientists to share models. By default, the DSR includes popular tools like Apache Spark (with SparkR) and Python for distributed compute and machine learning programming, Apache Hive and Apache Drill for SQL, Apache Pig, Apache Zeppelin, and eight out-of-the-box visualization libraries. Users can easily load other visualization libraries via a pluggable framework. In addition, the Docker file is available so users can customize the DSR for their own specific applications needs. For example, if needed, users can install TensorFlow with a single command.

The Data Science Refinery improves **collaboration** among data science teams because it leverages Apache Zeppelin notebooks for automatic model sharing and sharing visualizations of results. The MapR platform is ideal for storing model and notebook repositories. That coupled with MapR's global namespace and superior replication capabilities makes collaborations easy and encourages the goal-focused communication needed for a DataOps style of work.

Agility and flexibility in analytics, AI and machine learning projects on MapR is boosted by the excellent pairing of Kubernetes orchestration of containerized applications with the MapR platform for data persistence. This combination ensures the flexibility of being able to run applications side-by-side in different environments and using different tools while accessing the same data. This results in an efficient and cost-effective multi-tenancy that is a great benefit for development and for production systems. In addition, data scientists can use MapR streams to create real-time machine learning pipelines, making it possible to meet the needs of the microcycles of modern business goals.

Out-of-box secure native access to all platform data is another key feature of the Data Science Refinery. The MapR Data Platform is secure by default, and the Apache Zeppelin pre-configured in the MapR DSR leverages and integrates with the platform security using built-in capabilities of the MapR Persistent Application Container that houses the DSR.

Together all these capabilities support superior data science both at the technical level and at the human level to make it easier to get value from AI and other data science applications.


MAPR