Harvard Business Review

REPRINT H03Z0H
PUBLISHED ON HBR.ORG
OCTOBER 20, 2017

ARTICLE
ANALYTICS

How to Spot a Machine Learning Opportunity, Even If You Aren't a Data Scientist

by Kathryn Hume

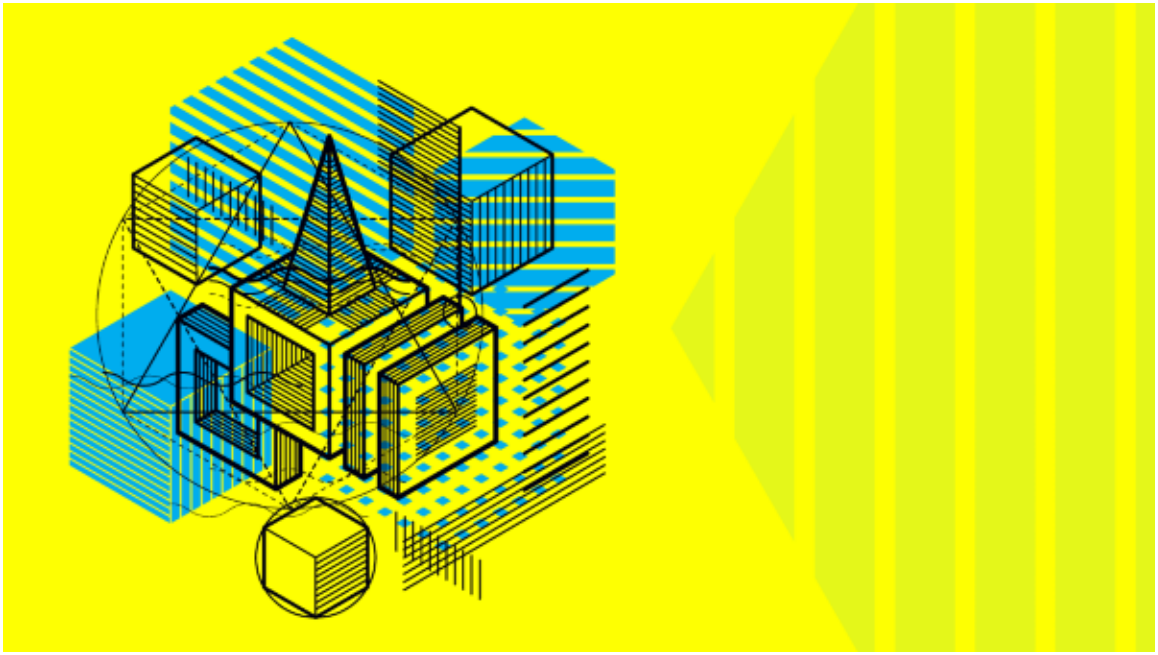
Harvard Business Review

ANALYTICS

How to Spot a Machine Learning Opportunity, Even If You Aren't a Data Scientist

by Kathryn Hume

OCTOBER 20, 2017



SYLVERARTS/ISTOCK

Artificial intelligence is no longer just a niche subfield of computer science. Tech giants have been using AI for years: Machine learning algorithms power Amazon product recommendations, Google Maps, and the content that Facebook, Instagram, and Twitter display in social media feeds. But

William Gibson's adage applies well to AI adoption: The future is already here, it's just not evenly distributed.

The average company faces many challenges in getting started with machine learning, including a shortage of data scientists. But just as important is a shortage of executives and nontechnical employees able to spot AI opportunities. And spotting those opportunities doesn't require a PhD in statistics or even the ability to write code. (It will, spoiler alert, require a brief trip back to high school algebra.)

Having an intuition for how machine learning algorithms work – even in the most general sense – is becoming an important business skill. Machine learning scientists can't work in a vacuum; business stakeholders should help them identify problems worth solving and allocate subject matter experts to distill their knowledge into labels for data sets, provide feedback on output, and set the objectives for algorithmic success.

As Andrew Ng has written: "Almost all of AI's recent progress is through one type, in which some input data (A) is used to quickly generate some simple response (B)."

But how does this work? Think back to high school math — I promise this will be brief — when you first learned the equation for a straight line: $y = mx + b$. Algebraic equations like this represent the relationship between two variables, x and y. In high school algebra, you'd be told what m and b are, be given an input value for x, and then be asked to plug them into the equation to solve for y. In this case, you start with the equation and then calculate particular values.

Supervised learning reverses this process, solving for m and b, given a set of x's and y's. In supervised learning, you start with many particulars — the data — and infer the general equation. And the learning part means you can update the equation as you see more x's and y's, changing the slope of the line to better fit the data. The equation almost never identifies the relationship between each x and y with 100% accuracy, but the generalization is powerful because later on you can use it to do algebra on new data. Once you've found a slope that captures a relationship between x and y reliably, if you are given a new x value, you can make an educated guess about the corresponding value of y.

As you might imagine, many exciting machine learning problems can't be reduced to a simple equation like $y = mx + b$. But at their essence, supervised machine learning algorithms are also solving for complex versions of m, based on labeled values for x and y, so they can predict future y's from future x's. If you've ever taken a statistics course or worked with predictive analytics, this should all sound familiar: It's the idea behind linear regression, one of the simpler forms of supervised learning.

To return to Ng's formulation, supervised learning requires you to have examples of both the input data and the response, both the x's and the y's. If you have both of those, supervised learning lets

you come up with an equation that approximates that relationship, so in the future you can guess y values for any new value of x.

So the question of how to identify AI opportunities starts with asking: What are some outcomes worth guessing? And do we have the data necessary to do supervised learning?

For example, let's say a data scientist is tasked with predicting real estate prices for a neighborhood. After analyzing the data, she finds that housing price (y) is tightly correlated to size of house (x). So, she'd use many data points containing both houses' size and price, use statistics to estimate the slope (m), and then use the equation y = mx + b to predict the price for a given house based on its size. This is linear regression, and it remains incredibly powerful.

Organizations use similar techniques to predict future product sales, investment portfolio risk, or customer churn. Again, the statistics behind different algorithms vary in complexity. Some techniques output simple point predictions (We think y will happen!) and others output a range of possible predictions with affiliated confidence rates (There's a 70% chance y will happen, but if we change one assumption, our confidence falls to 60%).

These are all examples of prediction problems, but supervised learning is also used for classification.

Classification tasks clump data into buckets. Here a data scientist looks for features in data that are reliable proxies for categories she wants to separate: If data has feature x, it goes into bucket one; if not, it goes into bucket two. You can still think of this as using x's to predict y's, but in this case y isn't a number but a type.

Organizations use classification algorithms to filter spam, diagnose abnormalities on X-rays, identify relevant documents for a lawsuit, sort résumés for a job, or segment customers. But classification gains its true power when the number of classes increases. Classification can be extended beyond binary choices like "Is it spam or not?" to include lots of different buckets. Perception tasks, like training a computer to recognize objects in images, are also classification tasks, they just have many output classes (for example, the various animal species names) instead of just Bucket 1 and Bucket 2. This makes supervised learning systems look smarter than they are, as we assume their ability to learn concepts mirrors our own. In fact, they're just bucketing data into buckets 1, 2, 3…n, according to the "m" learned for the function.

So far, this all feels rather abstract. How can you bring it down to earth and learn how to identify these mathematical structures in your everyday work?

There are a few ways you can determine whether a task presents a good supervised learning opportunity.

First, write down what you do in your job. Break apart your activities into: things you do daily or regularly versus things you do sporadically; things that have become second nature versus things that require patient deliberation or lots of thought; and things that are part of a process versus things you do on your own.

For those tasks that you perform regularly, on your own, and that feel automatic, identify how many others in your organization do similar tasks and how many people have done this historically.

Examine the nature of the task. Does it include predicting something or bucketing something into categories?

Ask yourself: If 10 colleagues in your organization performed the task, would they all agree on the answer? If humans can't agree something is true or false, computers can't reliably transform judgment calls into statistical patterns.

How long have people in the organization been doing something similar to this task? If it's been a long time, has the organization kept a record of successfully completed tasks? If yes, this could be used as a training data set for your supervised learning algorithm. If no, you may need to start collecting this data today, and then you can keep a human in the loop to train the algorithm over time.

Next, sit down with a data science team and tell them about the task. Walk them through your thought process and tell them what aspects of information you focus on when you complete your task. This will help them determine if automation is feasible and tease out the aspects of the data that will be most predictive of the desired output.

Ask yourself, if this were automated, how might that change the products we offer to our customers? Ask, what is the worst thing that could happen to the business if this were to be automated? And finally, ask, what is the worst thing that could happen to the business if the algorithm outputs the wrong answer or an answer with a 65% or 70% accuracy rate? What is the accuracy threshold the business requires to go ahead and automate this task?

Succeeding with supervised learning entails a shift in the perspective on how work gets done. It entails using past work — all that human judgment and subject matter expertise — to create an algorithm that applies that expertise to future work. When used well, this makes employees more productive and creates new value. But it starts with identifying problems worth solving and thinking about them in terms of inputs and outputs, x's and y's.

Kathryn Hume is vice president of product and strategy at integrate.ai, a Toronto-based startup.