David Chappell

INTRODUCING AZURE MACHINE LEARNING

A GUIDE FOR TECHNICAL PROFESSIONALS

Sponsored by Microsoft Corporation

Contents

What is Machine Learning?

Data can hold secrets, especially if you have lots of it. With lots of data about something, you can examine that data in intelligent ways to find patterns. And those patterns, which are typically too complex for you to detect yourself, can tell you how to solve a problem.

This is exactly what machine learning does: It examines large amounts of data looking for patterns, then generates code that lets you recognize those patterns in new data. Your applications can use this generated code to make better predictions. In other words, machine learning can help you create smarter applications.

For example, suppose you want to create software that can determine, with a high degree of accuracy, whether a credit card transaction is fraudulent. What's the right approach for doing this? One option is to get a few smart people together in a room and think about it, then write code that implements whatever they come up with. This is probably the most common approach to creating software solutions today, and it certainly can work.

But if there's data available about the problem you're trying to solve, you might instead use that data to figure out an effective solution. For example, suppose you're trying to find the best approach for detecting credit card fraud, and all you have to work with is the historical data shown in Figure 1.

Name	*Amount*	*Fraudulent*
Smith	$2,600.45	No
Potter	$2,294.58	Yes
Peters	$1,003.30	Yes
Adams	$8,488.32	No

Figure 1: With just a small amount of data, it's hard to find accurate patterns.

The good thing about having so little data is that you might be able to find a pattern just by looking at it. The bad thing about having so little data is that the pattern you find is likely to be wrong. Given the data in Figure 1, for example, you might decide that anybody whose name starts with "P" is a fraudster, which probably isn't correct.

With more data, your odds of finding a more accurate pattern get better, but finding that pattern will be more difficult. For instance, suppose you have the set of credit card transaction data shown in Figure 2 to work with.

Name	Amount	Where Issued	Where Used	Age of Cardholder	Fraudulent
Smith	$2,600.45	USA	USA	22	No
Potter	$2,294.58	USA	RUS	29	Yes
Peters	$1,003.30	USA	RUS	25	Yes
Adams	$8,488.32	FRA	USA	64	No
Pali	$200.12	AUS	JAP	58	No
Jones	$3,250.11	USA	RUS	43	No
Hanford	$8,156.20	USA	RUS	27	Yes
Marx	$7,475.11	UK	GER	32	No
Norse	$540.00	USA	RUS	27	No
Edson	$7,475.11	USA	RUS	20	Yes

Figure 2: More data can help in finding better patterns.

With this much data, it's immediately obvious that our first guess at a pattern—everybody whose name starts with "P" is a crook—can't be right. Look at the customer named Pali, for example. But if that simple guess was wrong, what's the right answer? In the data shown here, customers in their 20s seem to have a high fraud rate, but then look at customer Smith—he doesn't fit this pattern. Or could it be customers whose credit card was issued in the USA but used in Russia? That also looks suspicious. But no, look at the customer named Jones; she doesn't fit this pattern. Okay, then maybe a combination of things can be used to detect fraud. How about customers whose card was issued in the USA, used in Russia, and are in their 20s? Wrong again—the cardholder named Norse violates this rule.

The truth is that the pattern the data supports is this: A transaction is fraudulent if the cardholder is in his 20s, the card was issued in the USA and used in Russia, and the amount is more than $1,000. With some time, you probably would have figured this out, since the data you have to work with isn't very large.

But suppose you have not just ten records to work with, as in Figure 2, but ten million. And suppose that for each record, you have not just the six columns of data shown in Figure 2, but 60 columns. There's probably a useful pattern hidden in that data for determining which transactions are likely to be fraudulent, but you'll never figure it out by manually looking at the data—there's too much of it. Instead, you have to use statistical techniques, approaches that are designed for finding patterns in large amounts of data. And you'll need to apply those techniques to the data by running them on a computer.

This is exactly what the machine learning process does: It applies statistical techniques to large amounts of data, looking for the best pattern to solve your problem. It then generates an implementation—code—that can recognize that pattern. This generated code is referred to as a *model*, and it can be called by applications that need to solve this problem. With fraud detection, for example, the calling application must provide the right information about a transaction, such as age, amount, where the card was issued, and where it's being used. The model created through machine learning then returns an indication of whether this transaction is likely to be fraudulent.

And while fraud detection is a good example, machine learning is applicable to much more than this. This technology can be used to predict an organization's future revenues, determine whether a mobile phone customer is likely to switch to another vendor, decide when a jet engine needs maintenance, recommend movies for customers, or anything else where lots of historical data is available. (Because machine learning helps predict the future, it's often included in the broader category of *predictive analytics*.) All that's needed is the data, machine learning software to learn from that data, and people who know how to use that software.

Azure Machine Learning (Azure ML) is a cloud service that helps people execute the machine learning process. As its name suggests, it runs on Microsoft Azure, a public cloud platform. Because of this, Azure ML can work with very large amounts of data and be accessed from anywhere in the world. Using it requires just a web browser and an internet connection.

Still, understanding what this technology does requires a deeper look at how machine learning really works. The next section gives an overview of the process.

The Machine Learning Process

Whether an organization uses Azure ML or another approach, the basic process of machine learning is much the same. Figure 3 shows how it typically looks.

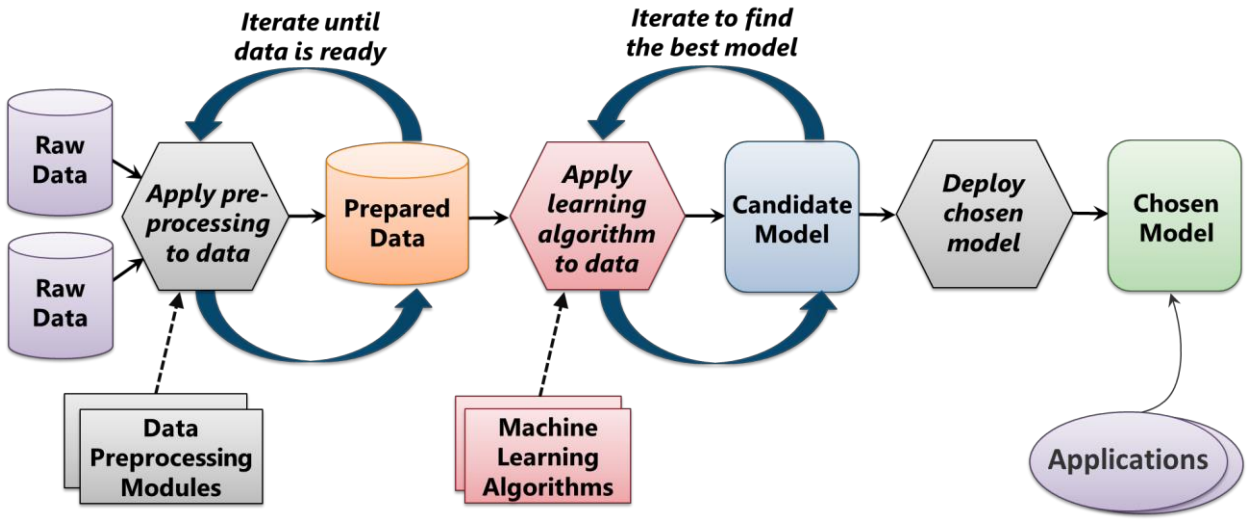

Figure 3: The machine learning process starts with raw data and ends up with a model derived from that data.

As the figure shows, machine learning starts with data—the more you have, the better your results are likely to be. Because we live in the big data era, machine learning has become much more popular in the last few years. Having lots of data to work with in many different areas lets the techniques of machine learning be applied to a broader set of problems.

Since the process starts with data, choosing the right data to work with is critically important. For example, what data should we use to detect credit card fraud? Some things are obvious, such as a customer's age, the country in which the card was issued, and the place the card is being used. But other data might also be relevant, such as the time of day the card is being used, the kind of establishment it's being used in, and maybe even the weather at the

time of use. Determining the most relevant data for a machine learning project is a fundamental part of the process, and it's an area where domain experts usually have a lot to offer. Who better to decide what the likely indicators of fraud are than businesspeople who work in this area?

Whatever data you choose, it's often not in the right condition to be used directly. Instead, machine learning projects typically process that raw data with various data preprocessing modules, as Figure 3 shows. With credit card fraud detection, for example, the raw data might contain duplicate entries for some customers, perhaps with conflicting information. Or the raw data might contain holes, such as a lack of information about where some cards were issued or used.

The goal of data preprocessing is to create what's called *prepared* data. But getting here usually isn't simple. As Figure 3 shows, it's an iterative process, with several different data preprocessing modules applied to the raw data. In fact, choosing the best raw data to start with, then creating prepared data from that raw data frequently takes up the majority of the total time spent on a machine learning project.

Once a machine learning team has the right prepared data, it can move on to the next step: searching for the best way to solve the problem they're working on, such as detecting credit card fraud. To do this, the team uses *machine learning algorithms* to work with the prepared data. These algorithms typically apply some statistical analysis to the data. This includes relatively common things, such as a regression, along with more complex approaches, including algorithms with names such as two-class boosted decision tree and multiclass decision jungle. The team's *data scientist* chooses a machine learning algorithm, decides which aspects of the prepared data should be used, then examines the result. The goal is to determine what combination of machine learning algorithm and prepared data generates the most useful results. For example, if the goal is to determine whether a credit card transaction is fraudulent, the data scientist chooses the parts of the prepared data and an algorithm that she thinks are most likely to accurately predict this.

As mentioned earlier, when a machine learning algorithm is run on prepared data, the result is referred to as a model. A model is code—it's the implementation of an algorithm for recognizing a pattern, e.g., determining whether a credit card transaction is fraudulent. But don't be confused: A machine learning algorithm and the algorithm implemented by a model are different things. The machine learning algorithm is run over prepared data, and the goal is to create a model. The algorithm implemented by the model itself provides a solution that actually solves a problem. Models are accessed by applications to answer questions like "Is this transaction fraudulent?" Machine learning algorithms are used only during the machine learning process itself.

It's also important to understand that a model typically doesn't return a yes-or-no answer. Instead, it returns a probability between 0 and 1. Deciding what to do with this probability is usually a business decision. For example, if a credit card fraud model returns a probability of 0.9, this will likely result in the transaction being marked as fraudulent, while a probability of 0.1 will let the transaction be processed normally. But what if the model returns a fraud probability of 0.5? Should the transaction be accepted, or should it be blocked as fraudulent? Answering this question is a business decision. Is it more important to catch fraudulent transactions, even at the risk of giving a good customer a bad experience? Or is it better to accept some level of fraud to avoid annoying honest customers? These are questions that data scientists and developers can't typically answer—they're business issues.

In pretty much every case, the first candidate model created isn't the best one. Instead, the data scientist will try many different combinations of machine learning algorithms and prepared data, searching for the one that produces the best model. Each of these attempts (i.e., each iteration) can be thought of as an *experiment*, and typical projects will run many experiments during the search for the right model, as Figure 3 suggests.

Once an effective model exists, there's one more step: deploying that model. This allows applications to use the algorithm the model implements. This is an important part of the process—without deployment, all of the work done so far is worthless. And if deployment is too difficult, the people who create the model might not be able to deploy new models quickly, something that's often required in our rapidly changing world.

However it's deployed, a model implements an algorithm for recognizing patterns. And where did that model come from? It was derived from the data. Rather than putting a few smart people in a room and letting them invent a way to solve a problem, machine learning instead generates an effective solution from data. When you have lots of data to work with, this can be a very effective approach.

What is a Data Scientist?

Many aspects of the machine learning process aren't simple. For example, how should you determine which raw data to use? How should that data be processed to create prepared data? And perhaps most important, what combinations of prepared data and machine learning algorithms should you use to create the best model? There's a substantial amount of complexity here that someone has to deal with.

Complex problems are usually solved by specialists, and so it is with machine learning. A data scientist is a specialist in solving problems like the ones that arise in machine learning. People in this role typically have a wide range of skills. They're comfortable working with complex machine learning algorithms, for example, and they also have a strong sense of which of these algorithms are likely to work best in different situations. A data scientist might also need to have software development skills, as they're often called upon to write code.

Perhaps the most important fact about data scientists, though, is that pretty much every machine learning project needs one. Without a specialist to guide a team through this thicket of complexity, the team is quite likely to get lost.

What Azure ML Provides

The machine learning process isn't especially simple. To make life easier for people doing machine learning, Azure ML provides several different components. Figure 4 shows what those components are and where they fit into the machine learning process.

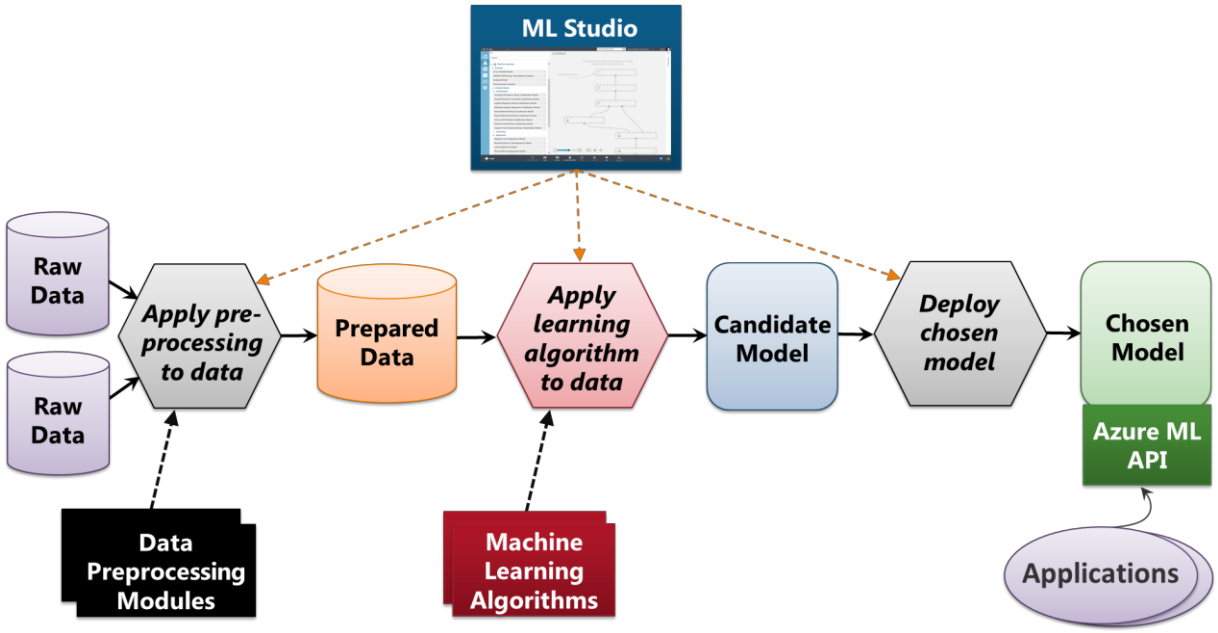

Figure 4: Azure ML provides a graphical tool for managing the machine learning process, a set of data preprocessing modules, a set of machine learning algorithms, and an API to expose a model to applications.

The components that Azure ML provides are the following:

- *ML Studio*, a graphical tool that can be used to control the process from beginning to end. Using this tool, people on the machine learning team can apply data pre-processing modules to raw data, run experiments on the prepared data using a machine learning algorithm, and test the resulting model. Once an effective model is found, ML Studio also helps its users deploy that model on Microsoft Azure.
- A set of data preprocessing modules.
- A set of machine learning algorithms.
- An Azure ML API that lets applications access the chosen model once it's deployed on Azure.

All of these are important, and we'll examine them a bit more closely later. For now, however, it's worth looking in a little more detail at ML Studio. Figure 5 shows an example of the user interface this tool provides.

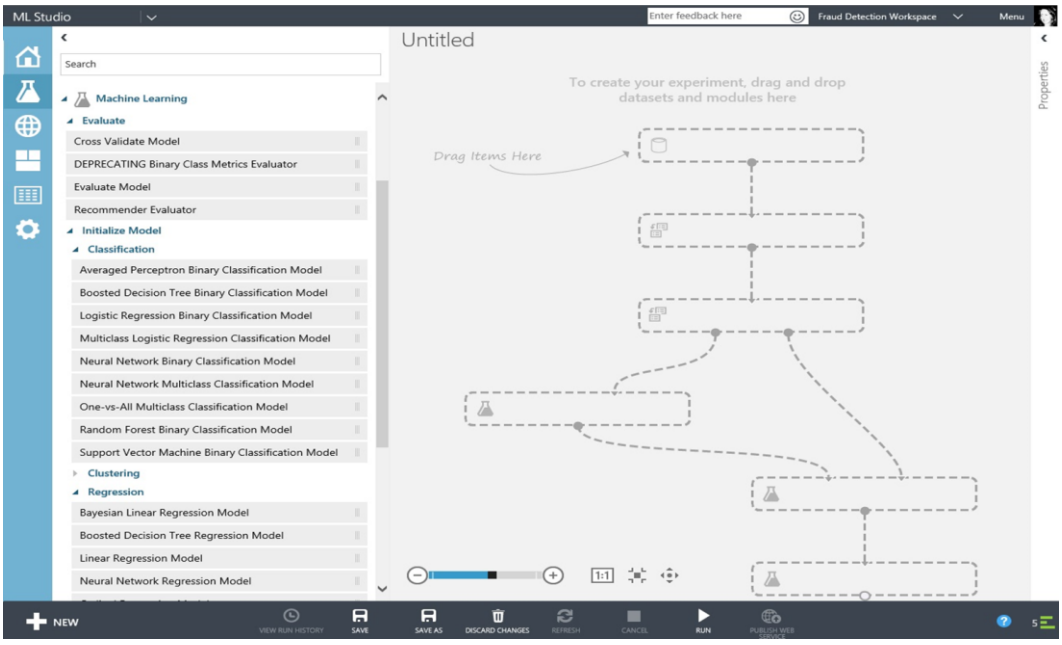

Figure 5: ML Studio is a graphical tool that supports the entire machine learning process, from preprocessing data to deploying a model.

As the figure suggests, ML Studio lets a user drag and drop datasets, data preprocessing modules, machine learning algorithms and more onto its design surface. The user can connect these together graphically, then execute the result. For example, a data scientist might use ML Studio to connect a dataset that holds prepared data with the machine learning algorithm he's chosen for a particular experiment. Once this is done, he can use ML Studio to run the experiment and evaluate the model it creates. When he has what he believes is the best possible model, he can use ML Studio to deploy this model to Microsoft Azure, where applications can use it. Rather than relying on ad hoc, largely manual mechanisms to do all of this, ML Studio provides a single tool for controlling the entire machine learning process.

A Closer Look at Azure ML

It's important to understand the big picture of machine learning. If you'll be working alongside data scientists on a machine learning project, however, you need to know more than this. Like most technologies, machine learning, has its own specialized jargon, terms that can be a little confusing. What follows looks a bit more deeply at the terminology and technology of machine learning through the lens of Azure ML.

Preparing Data

As mentioned earlier, getting data in shape for machine learning algorithms to use can be the most time-consuming part of a machine learning project. A look at Figure 6 helps see why this is true.

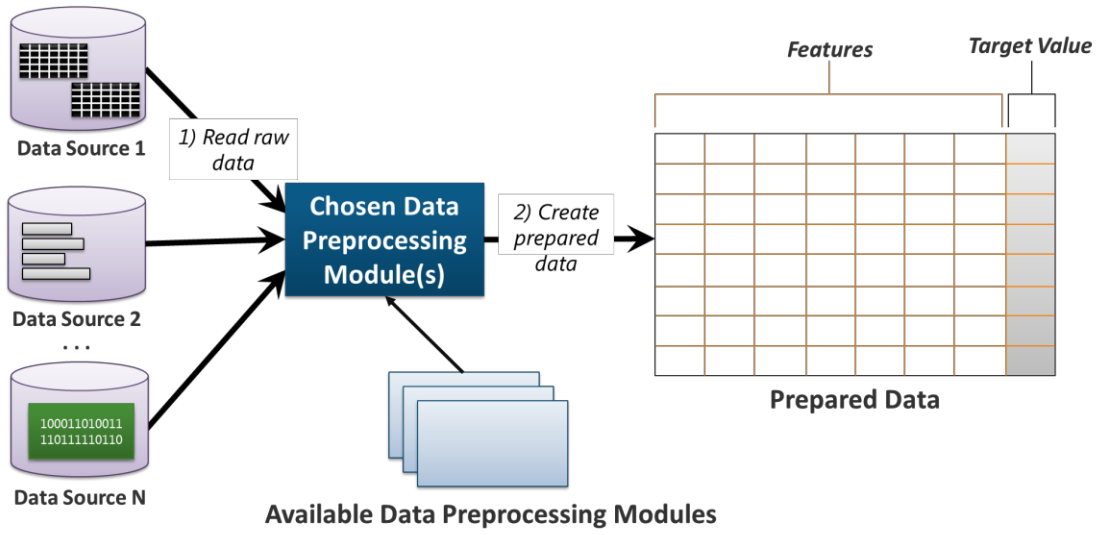

Figure 6: Data preprocessing turns raw data into a table, with each column in the table called a *feature*.

As the figure shows, prepared data is stored as a table. If you're lucky, the raw data you're working with is already stored in a table, such as with a relational database system. If you're not, the raw data might be stored in several different ways. The options include NoSQL stores, such as Azure's DocumentDB, binary data stored in Hadoop, and more. To transform this raw data into prepared data requires reading the raw data from multiple data sources (step 1), then running it through various data preprocessing modules to create prepared data (step 2). All of these things are done using ML Studio.

Azure ML provides a variety of data preprocessing modules. The choices include the following:

- *Clean Missing Data*, which lets an Azure ML user fill in missing values in a dataset. The user might choose to replace all missing values with zero, for instance, or replace each missing value with the mean, median, or mode of the other values in this column in the table.
- *Project Columns*, which removes columns that aren't useful. For example, two columns in the table might contain data that are very highly correlated with each other. In this situation, the machine learning process needs only one of the columns. This data preprocessing module lets its user delete the columns in a dataset that aren't useful.
- *Metadata Editor*, which allows changing the types of columns in a dataset.
- *Apply Math Operations* that allows performing mathematical operations on data. For example, suppose the raw data has a column containing prices in dollars and cents. If an Azure ML user needs only values in dollars, she could use this module to divide every value in that column by 100, then treat the result as dollars.

It's also possible to use other data preprocessing modules written in R or Python. Azure ML doesn't limit a user's choices to only modules provided by the service itself.

In a prepared data table, each column is called a *feature*. Often (although not always, as described later), one of the table's columns contains the value we're trying to predict. For example, the prepared data used to determine

whether a credit card transaction is fraudulent will typically be information about a large number of previous transactions, with each row containing the information for one transaction. Each row will also contain an indication of whether this particular transaction was or was not fraudulent, as illustrated back in Figures 1 and 2. This value is the thing we're trying to predict, and so it's called the *target value*, as Figure 6 shows.

The Role of R

Much of the work done by a data scientist involves statistics. For example, machine learning algorithms commonly apply some kind of statistical technique to prepared data.

But doing this kind of work can sometimes require programming. What programming language is best for statistical computing? The answer is clear: It's the open-source language called *R*. Created in New Zealand more than 20 years ago, R has become the lingua franca for writing code in this area. In fact, it's hard to find a data scientist who doesn't know R.

Because of this, Microsoft has included broad R support in Azure ML. A data scientist can use it to implement new machine learning algorithms, new data preprocessing modules, and more. You don't need to know R to use Azure ML, but users who do won't have to learn a new language just for Azure ML.

Running Experiments

Once the data is ready, it's time for the part that most data scientists like best: running experiments. Each experiment runs one or more machine learning algorithms over some part of the prepared data. The result is a model, an implementation of an algorithm that the data scientist hopes will solve the problem at hand, e.g., an algorithm that correctly determines whether a credit card transaction is fraudulent. Figure 7 illustrates the process.

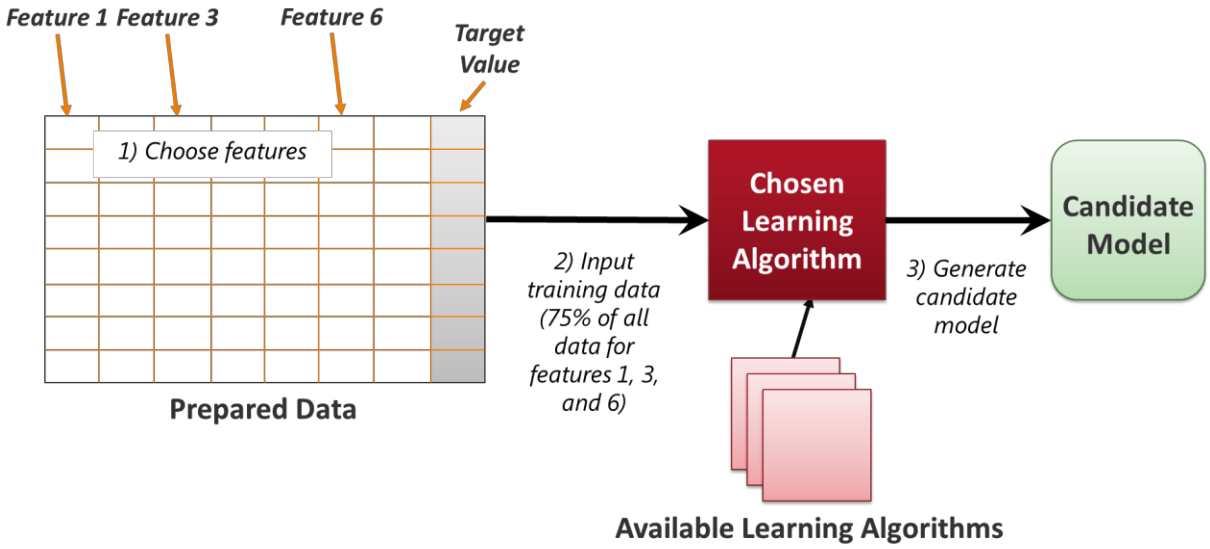

Figure 7: An experiment creates a candidate model from prepared data.

As the figure shows, the data scientist can choose which features (i.e., which columns) in the prepared data table should be used. In this example, the data scientist selected features 1, 3, and 6 (step 1). These features are then fed into whatever machine learning algorithm the data scientist has chosen for this experiment (step 2), creating a candidate model (step 3).

A couple of things are worth noting here. First, in the jargon of machine learning, the process of finding the best model is known as *training the model*. Accordingly, the input data used to create a candidate model is called *training* data. Second, notice that in step 2, only 75% of the data in the three chosen features is fed into the learning algorthim. 25% of the data is held back. Why is this? The answer is that the held-back data will be used to test this candidate model, something that's described in more detail later.

Machine Learning Algorithms

Azure ML provides a large set of machine learning algorithms, and data scientists are also free to create their own. For the most part, understanding and choosing the right algorithm is the province of data scientists—it's a job for specialists. Still, there are some basics that even non-data scientists can understand.

For example, some machine learning algorithms are especially good for *classification* problems. In problems like this, the goal is to create a model that can take a set of input data and accurately determine which of two or more pre-defined categories this input data falls into. A good example of this is the credit card fraud scenario we've been looking at throughout this paper. It is in fact a classification problem. The goal is to take data about a credit card transaction, then find a model that accurately classifies the transaction into one of two categories: valid or fraudulent. Azure ML provides several different algorithms for classification problems, with names like *Multiclass Decision Jungle*, *Two-Class Boosted Decision Tree*, and *One-vs-All Multiclass*. To use one of these algorithms, a user can just choose it from a dropdown menu provided by ML Studio.

Another group of machine learning algorithms are good for *regression* problems. For example, you might use regression to predict future demand for a product based on previous demand, the weather, and other things. To

do this, you need historical data about the relationship between demand (the target value) and the other input data (i.e., the features). Azure ML supports several different algorithms for doing this, including *Bayesian Linear Regression*, *Boosted Decision Tree Regression*, and *Neural Network Regression*. As with classification algorithms, a user specifies which one of these she'd like to use by choosing it from a dropdown menu in ML Studio.

Yet another group of machine learning algorithms is those used for *clustering* problems. Here, the goal is find meaningful clusters by examining large amounts of existing input data, then create a model that can figure out from new data about some item what cluster this new item should be assigned to. For example, a data scientist might use a clustering algorithm to determine that a firm's customers fall into three clusters: Loyal, Somewhat Loyal, and Likely to Switch. Whenever a customer contacts a call center, a model created using a clustering algorithm could determine which cluster that customer falls into. The call can then be routed to different customer service people based on the caller's cluster. Perhaps the customer service people who handle calls from Likely to Switch customers are empowered to offer this customer special deals to entice them to stay, while the customer service people who handle calls from Loyal customers aren't allowed to do this. Azure ML supports only a single clustering algorithm, called *K-Means Clustering*, although data scientists can add more if desired.

Supervised and Unsupervised Learning

Machine learning algorithms can be grouped into two flavors: supervised and unsupervised. With supervised learning, you know what target value you're trying to predict. The goal is to map some set of features into the correct target value for that row. Since you know the target values, you're doing *supervised* learning. Both classification and regression algorithms are used for supervised learning. The example used throughout this paper, detecting credit card fraud, is also supervised learning, because the value you're looking for (whether or not a transaction is fraudulent) is contained in the historical data you're working with.

With *unsupervised* learning, you don't know what target values you're looking for. Instead, the people running this project have to decide when they're done, i. e., when the model is producing good-enough results. An example of unsupervised learning is the scenario just described that clusters customers into three groups based on their loyalty. Clustering algorithms are used for unsupervised learning, and it's up to the people involved to decide whether the results of the clustering algorithm best support three groups of customers, four groups of customers, or something else. Unlike supervised learning, there's no clear target value that you're trying to predict.

As might be obvious, unsupervised learning problems are generally harder than those using supervised learning. Fortunately, a substantial majority of the most common machine learning scenarios today use supervised learning.

Testing a Model

Once a candidate model exists for this experiment, the next challenge is to determine how good the model is. Can it really take data from the chosen features and correctly predict the target value? To see how well the candidate model does this, we need to test it.

But where should the test data come from? The answer is simple: The test data comes from our prepared data table. Since we held back 25% of the data in the table when we created this candidate model, we can now use that held-back data to test how well our model performs. Figure 8 illustrates this process.

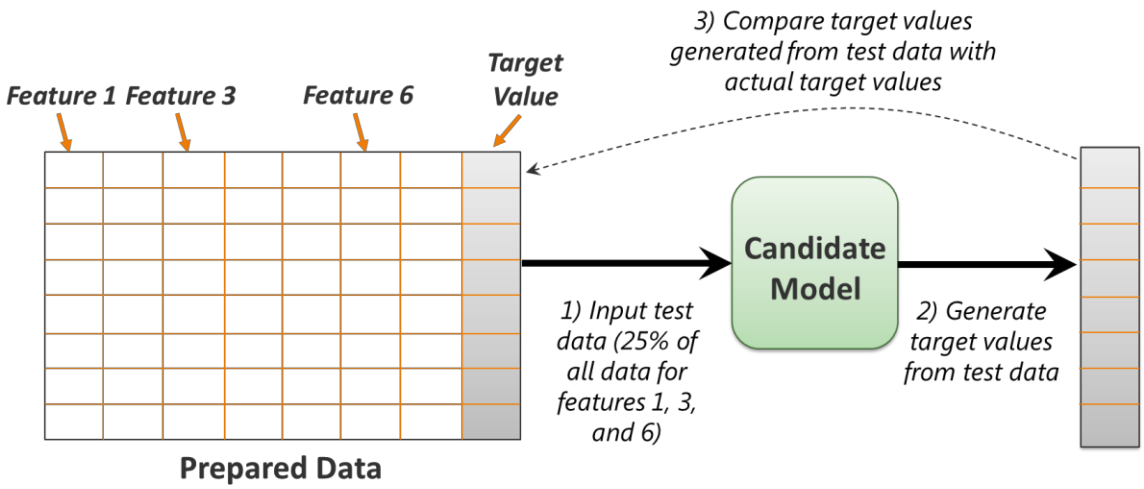

Figure 8: An experiment can compare the target values produced by the candidate model with real instances of the target value.

The process begins by feeding the test data into the candidate model (step 1) and letting that model generate target values from this data (step 2). The team working on this project can then compare the generated target values with the target values in the test data. If they match well, then we have a model that can be deployed and used. If not, the data scientist goes back to the drawing board, choosing a new combination of features (data) and machine learning algorithms.

As mentioned earlier, the process of creating a candidate model, then testing it with some held-back data is called training the model. It's just as accurate to think of the process as a series of experiments, though, each an educated guess about what data and learning algorithm are likely to generate the best model. In effect, "training the model" is the phrase that machine learning people use to describe what's really a trial-and-error process.

Azure ML has built in support for doing all of this. Using Studio ML, for instance, it's straightforward to split a dataset containing prepared data, use part of that data to create a model, then use the remaining part of the data to test the model. Rather than doing all of this manually or with ad hoc tools, Azure ML provides a consistent approach to creating and testing models. And to make it easier for groups working together in different places, Azure ML wraps all of the information about a specific machine learning project into a workspace, then lets that workspace be accessed by different people working in different places. Because Azure ML runs in the public cloud, sharing a workspace is simple.

Deploying and Using a Model

Once the people working on a machine learning project have a model they're happy with, they need a way to deploy that model so it be accessed by applications. Deployment can be the Achilles' heel of machine learning, as it's traditionally been done manually. And models have often been created in R, without a straightforward API to access the hard-won algorithm the model implements. Yet to be really effective, models often need to be recreated—and redeployed—frequently. A firm in a fast-moving market might need to roll out a new model as often as every week. If deployment is slow and painful, frequent deployment isn't possible, which can leave businesses stuck with old models.

Azure ML changes this. In fact, one of the strongest aspects of this machine learning technology is its support for fast, frequent deployment of models through an Azure ML API service. Figure 9 shows how this looks.

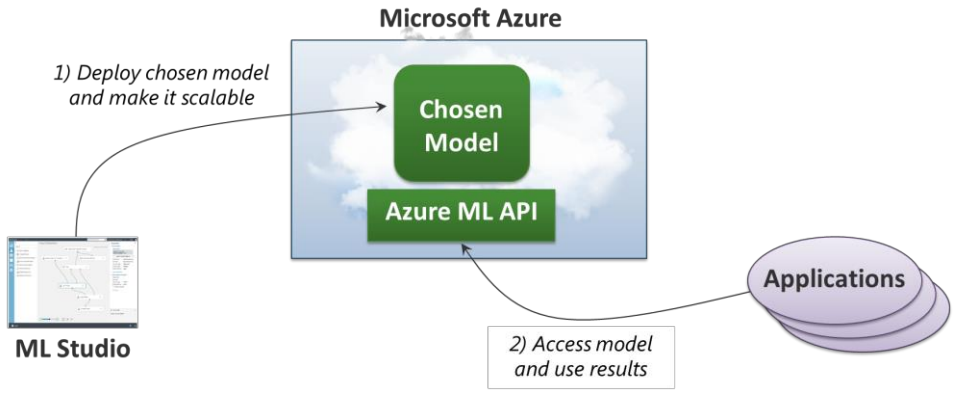

Figure 9: Azure ML can deploy a model directly into Microsoft Azure, where applications access it through a RESTful interface.

Once a model is ready, a member of the team can use ML Studio to deploy it directly to Microsoft Azure (step 1). The model exposes its functionality through the Azure ML API's RESTful interface, which lets applications access this model and use the results it provides (step 2). The API exposes both a synchronous service for requesting single predictions and an asynchronous service for sending in bulk requests.

As the figure suggests, Azure ML does more than just deploy a model. It also automatically sets up the model to work with Azure's load balancing technology. This lets the model grow to handle a large number of client applications, then shrink when demand falls. And although it's not shown in the figure, Azure ML first deploys a model to a staging area, where it can be tested. Once the model is ready, its owner can use ML Studio to move it into production.

It's also possible to publish a model's API into the Microsoft Azure Marketplace. This makes your model available to other users on the Internet, giving you a way to monetize your work. In fact, Microsoft itself provides some Azure ML-based services in the Marketplace, such as a service for predicting customer churn.

How applications actually use a model is worth a little more discussion. Figure 10 shows what's really going on.

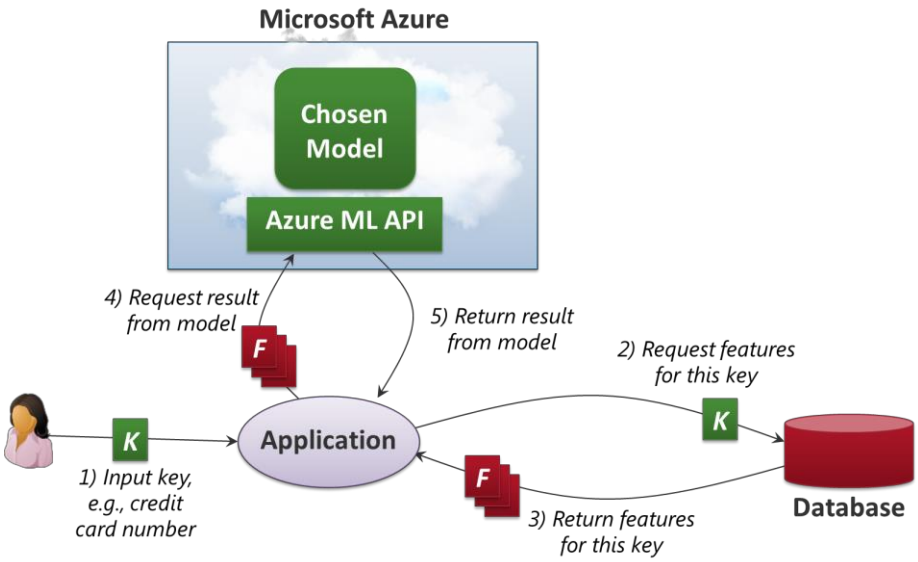

Figure 10: Applications must provide data for the features required by a model.

To get the right result from a model, the application must provide the right inputs. In other words, the application must provide values for the features that the model needs to determine the correct output. If the model detects credit card fraud, for example, the output is typically a probability that a transaction is fraudulent, while the inputs include things such as the customer's credit card number, the country in which the card was issued, the country in which it's being used, and other features required by this model.

In fact, a model might require hundreds or even thousands of input values, one for each feature. Where does the application get all of this data? One possible approach is illustrated in Figure 10. Here, the application gets a simple key, such as the user's credit card number (step 1), then uses that key to look up values for the necessary features in a database (step 2). The database returns values for those features (step 3), and the application then includes those features in its request to the model (step 4). The value that comes back (step 5) is the result of the model's evaluation of those features, e.g., the probability that this transaction is fraudulent.

Pricing

Anybody who wants to use Azure ML needs to understand how that service is priced. For Azure ML, Microsoft provides two pricing tiers:

- A free tier, which allows using a large subset of the service with up to 10 gigabytes of data, but without the ability to deploy models into production. This tier is intended to let potential customers kick Azure ML's tires, getting a feeling for what the service offers and what using it is like.
- A standard tier, which allows access to everything Azure ML provides and the ability to use larger data sets. This tier also provides a service level agreement (SLA) for Azure ML.

Pricing for the standard tier has several components:

- Each user of Azure ML must have a subscription to the service, paying a per-seat, per-month fee. Note that only users of Azure ML itself must have subscriptions. There's no subscription fee required for people who use applications that rely on Azure ML models.
- Along with a subscription, each Azure ML user also pays a fixed amount per hour for using ML Studio. This fee only applies to active use of ML Studio; a user doesn't pay the fee when she's not using this cloud-based tool.
- For models in production, Microsoft charges for each compute hour the running model consumes, along with a small fee for each API call made to the model.

For current Azure ML prices, see http://azure.microsoft.com/en-us/pricing/details/machine-learning/.

Conclusion

The idea of machine learning has been around for quite a while. Because we now have so much more data, machine learning has become useful in more areas. Yet unless the technology of machine learning gets more accessible, we won't be able to use our big data to derive better solutions to problems, and thus build better applications.

A primary goal of Azure ML is to address this challenge by making machine learning easier to use. While data scientists are still required, this cloud service can help less-specialized people play a bigger role in bringing machine learning into the mainstream. Going forward, expect data-derived models to become more common components in new applications.

Most people already realize that this is the big data era—it's too obvious to ignore. Less obvious but perhaps just as important is this: The rise of big data means that this is also likely to be the machine learning era.

About the Author

David Chappell is Principal of Chappell & Associates (http://www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technologies.