



---

AN INTRODUCTION TO

# ACTIVE LEARNING

By Jennifer Prendki

VP of Machine Learning

Figure Eight

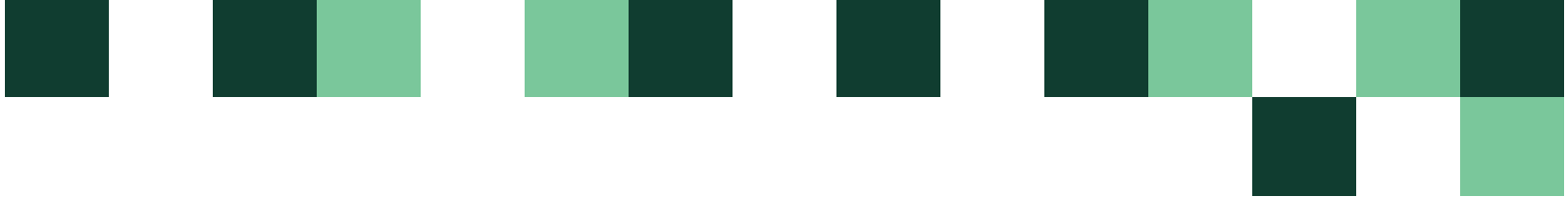
# INTRODUCTION

As the global availability of high-speed compute has grown and the AI winter has thawed, decades of theoretical approaches are being tested and proven. Breathless articles in everywhere from big label publications like Wired and the Harvard Business Review to the smaller publication data scientists have frequented for years have trumpeted these gains and with good reason.

AI is increasingly affecting our daily lives, preventing credit card fraud, aiding doctors with disease diagnosis, powering the smartest search results, keeping products on the shelves, predicting environmental disasters, and so much more.

But while the successes and impact of AI projects get the lion's share of publicity, what's often missed is the data that drives it all. After all, it's both the availability of compute resources (specifically GPUs) and the prevalence of high quality training data combined with smart learning schemas that create successful ML deployments.





Here at Figure Eight, we annotate a lot of the raw data that powers these AI solutions. And much of that data simply wasn't available before data storage became far more affordable during the Big Data era that started sometime in the 1990's.

Here's the thing, though: many organizations simply have too much data. The explosion of IoT devices paired with the increasingly prevalence of memory-greedy data formats like high resolution images and video have left some data scientists fending for themselves in an ocean of raw data.

Of course, a preponderance of data is a key ingredient to creating real-world machine learning, but it also poses a challenge for data scientists: unless at least some of that data is labeled, it's essentially useless for any ML approach that relies on supervised or semi-supervised learning. And when there's simply too much data to label in the first place, it leaves data scientists with a challenge: which data needs to be labeled? How much of a dataset needs to be labeled for an ML application to be viable? How can we solve the problem of having more data than we can reasonably analyze?

One promising answer lies in an approach that enjoys far less publicity and research than deep learning or other commonly practiced tactics: active learning.

Active learning is unique in that it can both solve this data labeling crisis and train models to be more accurate with less data overall. In our Introduction to Active Learning, we'll walk you through:

- The pros and cons of active learning as an approach
- The three major categories of active learning
- How your active learner should decide which rows need labeling
- How to obtain those labels
- How to tell if active learning is appropriate for your ML project

**First, however, we'd like to start by looking at the overarching issue of data labeling and how to solve budgetary or time-based constraints.**



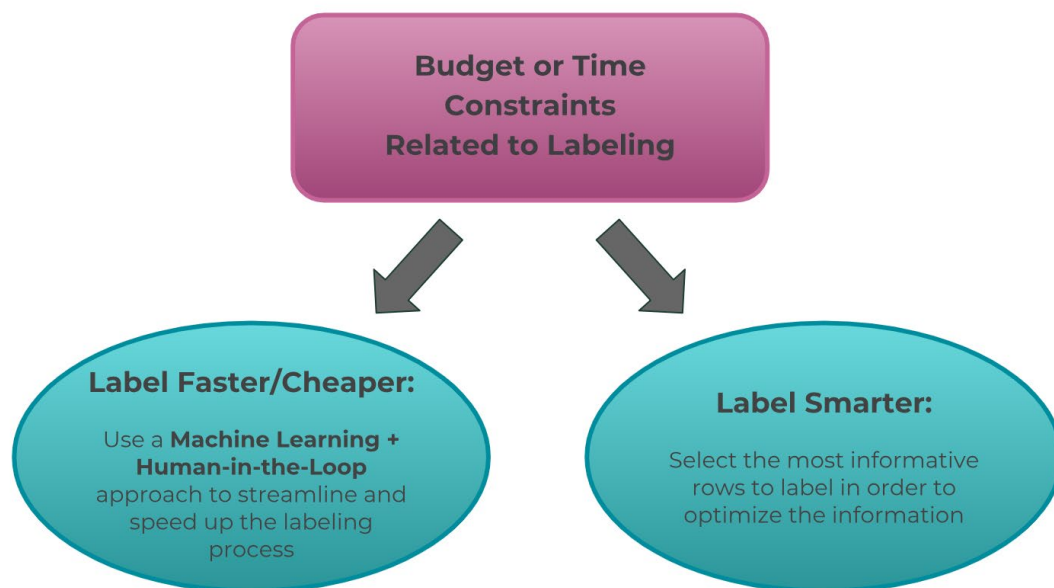
**LABELING FASTER**

**VS.**

**LABELING SMARTER**




Data scientists and machine learning practitioners don't have unlimited budget and time. But many do have what is a nearly unlimited supply of data.



Now, in a supervised learning context, most raw data needs some sort of labeling or augmentation. The question is how those labels are attained. A good strategy for this is balancing labeling quickly and labeling intelligently.

Let's say you're building an image classifier and have a dataset of unlabeled images. Those raw images need labels to train your model. "Labeling faster" means getting those quickly, but without breaking the bank.



For many data scientists (and many of our customers), that means leveraging human intelligence at scale for labels. Human annotators can look at a subset of your unlabeled images and provide the annotations you need to get model trained. This, of course, is a key step: you're supplying your own ontology, with your own classes, so your model will understand what you need it to understand and make the predictions that are important for your project. Our platform has been providing exactly these labels for over a decade. And there's a great deal of precedent for human

annotations training models, of course. ImageNet was built by labeling every image in that dataset, for just one example.

But again, given the budget and time constraints every practitioner deals with, there's only so much data that's reasonable to label. Oftentimes, you can't have humans label everything. That's where "labeling smarter" comes in.

Labeling smarter means identifying the unlabeled data in your original dataset that will provide the most value for your model.

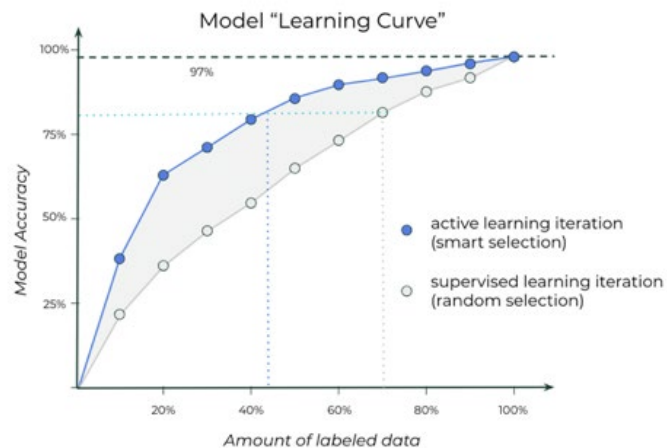
As data scientists, we've been trained to think that more data equals higher model accuracy. And while this is absolutely true, it's also important to acknowledge that not all data is created equally. Not all examples carry the same quality of information. Some data is going to be redundant. Identifying the best instances to train a model happens at two key times: before the model is even built and while the model is being trained. The former is called "prioritization." The latter is called "active learning."

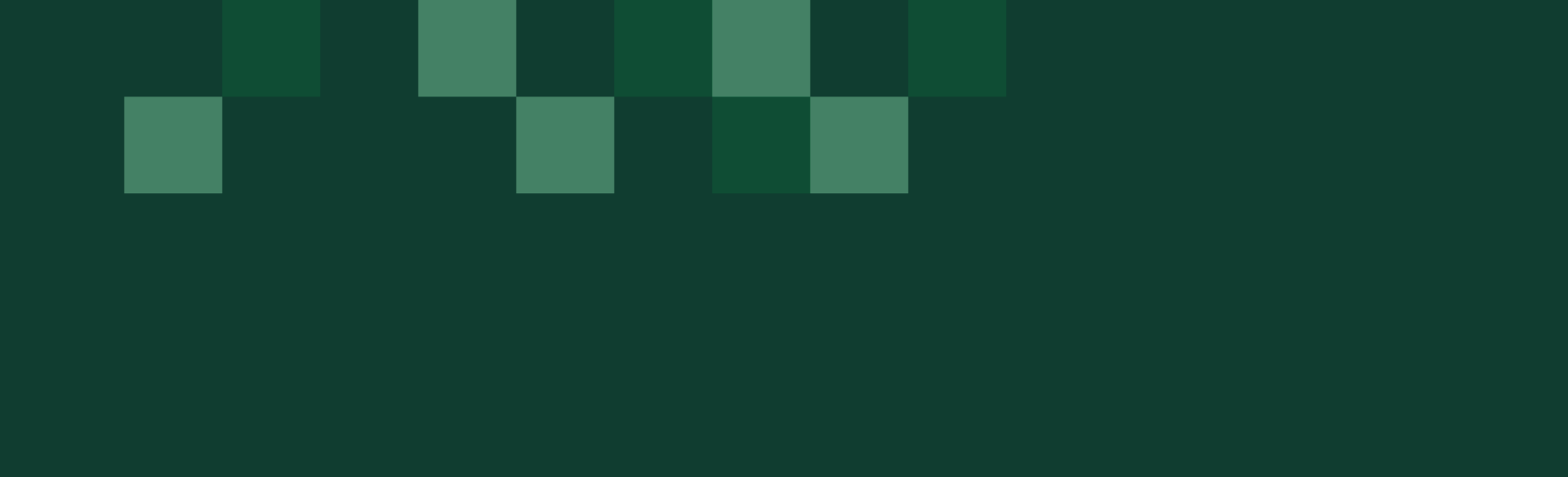
# WHAT IS ACTIVE LEARNING?

Active learning is a discipline of machine learning in which an algorithm proactively selects a subset of unlabeled data to be labeled next. In other words, from a pool of raw data, the algorithm identifies instances that will improve its accuracy.

This makes active learning an especially attractive for projects where labels are difficult, time-consuming, or expensive to collect. Put simply, algorithms like this—sometimes called “active learners”—can attain higher levels of accuracy with fewer labels than typical supervised models because they can identify the gaps in

their training and get labels the model needs to see to improve. It's also important to note here that active learning is not a replacement for models or model-building, but a way to enhance a model's accuracy and performance.





We'll discuss how this works in much more detail in the next few sections, but we'd like to start with a simple example to show how an active learning framework works generally. We'll use a simple image classifier as a guide, but active learning, of course, works for myriad use cases.

Now, say you're creating that simple image classifier. You start by selecting a random sampling of data to label and send it to oracles to provide those labels. But in this random training set, there happen to be a ton of labeled examples of dogs but very few of cats. When the model is looking at an unlabeled dataset, it behaves how you'd expect: because it's seen a lot of "dog" labels, it's fairly

confident predicting which images from the unlabeled set are dogs. But it's far less confident on those unlabeled cat images. It simply hasn't had enough training. The model hasn't converged.

What the model will do then is dynamically choose raw images it needs labeled to improve its accuracy. Unsurprisingly, this will mean it will not be asking for "dog" images and its confidence on that label is already high. These labels are generally provided by humans-in-the-loop, sometimes called "oracles" in this context, and fed back into the model to improve its accuracy.

In other words, active learning is a prime example of the marriage of human and machine intelligence. Humans provide the labels that train the model (labeling faster), the model decides what labels it needs to improve (labeling smarter), and humans again provide those labels. It's a virtuous circle that improves model accuracy faster than brute force supervised learning approaches, saving both time and money while creating a model that's real-world ready.

**Now, let's get into the weeds a little bit. Next, we're going to look a bit more deeply into how active learners really work. We'll start by laying out how these algorithms sample data.**





# HOW DOES



## ACTIVE




## LEARNING

## WORK?

In essence, deciding whether or not to query a specific label comes down to deciding whether the gain from obtaining that label offsets the cost of collecting that information. In practice, making that decision can take

several forms, depending in part on budget and restrictions your team might be under. The three scenarios we'll discuss here are stream-based selective sampling, pool-based sampling, and membership query synthesis.



# Pool-based approaches

This is probably the best known approach to active learning, so we'll start with pool-based sampling. Here, your active learner attempts to evaluate your entire dataset before selecting the unlabeled instances it needs annotated.

Usually, this means training your model on fully labeled, randomly sampled fraction of your data and generating an initial classifier. This classifier looks at your unlabeled dataset and makes queries for the instances it would like

to see labeled next (we'll get into the different approaches to this in the next section). Those instances are labeled by a human oracle and fed back into the model. Then the v2 of the model can select new instances it's uncertain or confused about and ask those be labeled next. This is a classic active learning loop and one we know well at Figure Eight, having been responsible for both training data creation and model tuning for myriad active learning projects.

Pooling is memory-greedy from a compute perspective, but it's great if you have a fixed budget, especially a fixed labeling budget. It's highly effective though, namely because pooling allows the active learning to select the best rows, not just the interesting ones that our next strategy will cover. There are, again, multiple ways an active learner will make this selection, but we'll get to those in a later section.

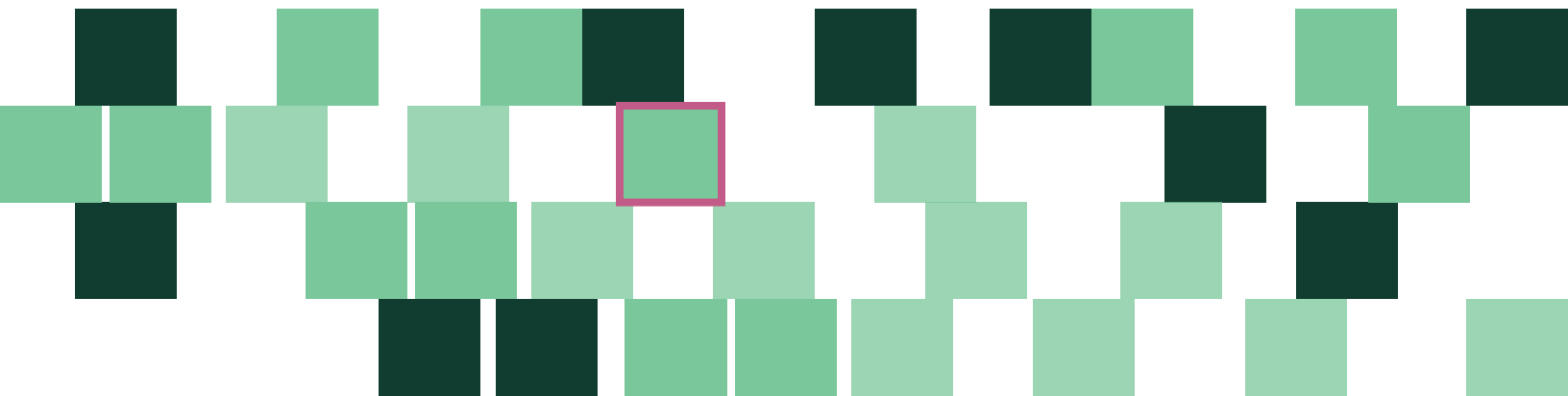


# Stream-based selective sampling

This scenario requires your model to determine if it would be beneficial to inquire for a specific label in your unlabeled, raw data pool as it's training. In other words, it's pulling data from a stream it needs to understand.

It works like this: as your model is being trained and it's presented a data instance, it decides "do I need to see this label?" It's looking for interesting unlabeled instances from a stream of unlabeled instances, but "interesting" to an active learning sometimes means it will be greedy and ask for more than it needs.

As such, this can be fairly costly and is most appropriate in instances where you have a good amount of budget to play with. But it can be truly valuable for when you're streaming data and can't afford to simply look at all of it holistically or when you're constrained on compute.

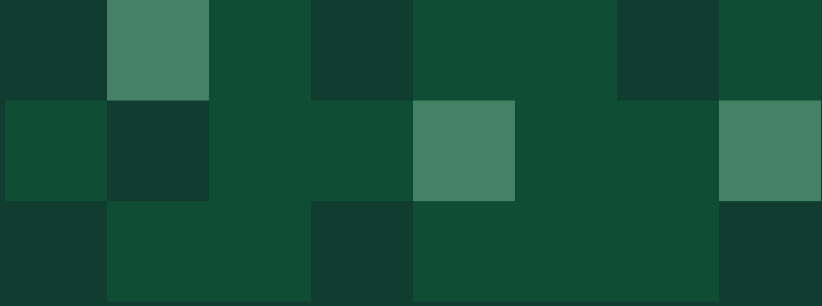


# Membership query synthesis scenario

This is more of an edge case and doesn't apply to all machine learning projects (or, indeed, even a significant percentage), but it can be a great tactic if you're hamstrung on unlabeled data or budget.

Here, the active learner actually generates its own synthetic data to be labeled. This is why it's not ideal for most situations (think image classifiers or most NLP projects). What it is good for however are domains like receipt transcription or, especially, search, where constructing legitimate synthetic data is far more feasible for a model that's begun to understand its space.

Again, typically, you'll see the pooling-based approach to active learning as it's much easier to constrain budget. So as we move through our next section, assume that's the strategy we're using—essentially, how the algorithm chooses from a pool of unlabeled data. Many of the below strategies can of course be used with streaming approaches, but for ease of explanation, we're going to stick with pooling.



# HOW DOES AN ACTIVE LEARNER DECIDE WHICH ROW TO LABEL?

There are four major querying strategies that active learning models use to determine the labels they need:

We'll discuss each in detail, starting with uncertainty sampling.

- Uncertain sampling
- Query-by-committee
- Expected impact
- Density-weighted methods





# Uncertainty sampling

This strategy is a lot like the image classifier example we gave above. Here, a model is first trained on a relatively small sample of labeled data, then applied to the unlabeled remainder. The algorithm chooses which instances it needs labeled over the next active learning loop based on the information it gained through this inference step. It does this in a few different ways.

The most popular is probably the least confidence strategy. Essentially, a model trained on the training portion of your initial dataset will then look at the unlabeled instances and ask that the ones it's **least confident** about be labeled next. Those labels get fed back into the model and, over several iterations, the model will improve its accuracy by understanding more labels.

To take it back for a moment, this is a prime example of the “labeling smarter” approach we mentioned a few sections ago. A by-product of the machine requesting labels first for the instances it's most unsure of is that you don't needlessly provide labels that it's already fairly confident about, which helps prevent both the accuracy paradox and cost overruns.

There's a potential drawback to this strategy which is that it can occasionally pollute your training data if your original data is particularly low quality.

Another common strategy for uncertainty sampling is known as **margin sampling**. Here, the model asks for annotations on instances where there are two likely labels but the margin between those labels is narrow. In other words, the model is predicting “this is either a dog or a cat but I can’t tell which.” Labeling those instances that are in a gray area can help the active learner identify the attributes that separate similar labels.

Both of the above examples of **uncertainty sampling** try to help the model discriminate between specific classes and, overall, they do a great job

reducing specific classification errors. However, there’s one additional strategy here we should highlight as well. It’s called **entropy based** sampling.

Entropy-based sampling doesn’t look for instances it’s the most unclear about or instances that it can’t quite tell apart. Instead, what it does is request labels for instances where the model has high output variance in its prediction. In other words, it’s not about the least confidence or parsing between like labels, it’s about variance and confusion.

And that actually underlines an important point here: you’ll notice that each of these uncertainty sampling scenarios is based on some kind of confusion in outcomes. Those instances—the ones about which the model needs additional input to increase its overall accuracy—get labeled by a human and fed back in. And in all of the above scenarios the expectation is, of course, that these additional labels improve the algorithm’s performance.



# Query by Committee (QBC)

Budget-wise, QBC is a more intensive process, generally requiring more compute than uncertainty sampling. But it's a powerful approach to active learning.

You again have your original labeled training data. But instead of using a single model and finding uncertain instances to label in order to improve the model's accuracy, you employ a committee of models, both on that original training data and as active learners.

Essentially, what's happening here is that you're looking for the highest disagreement on unlabeled data. In other words, where your various models look at an unlabeled data row and are especially at odds, those are the instances you'll want to label next. The core idea behind this framework is minimizing the version space.

After all, oftentimes, when data scientists develop ML solutions, they have more than one option at hand. And sometimes the choice is difficult, in particular because each type of algorithm presents some pros and cons. That's the reason why many scientists like to leverage ensemble methods: the idea to combine the output of several "weak" predictors (that don't even need to be really good) in order to create a strong one using the fact that the weaknesses of one model are offset by the strengths of the others.

Many ensemble methods (for example: a random forest) are voting systems: each one of the models built on the data has a "voting" right, and the final inferred value for a row is the one that most models agreed on (voted on). Similarly, query-by-committee is using a voting methodology. Simply put, the strategy is about querying the labels of the rows for which the highest level of disagreement was reached across the different models.





# Expected impact

This set of strategies attempts to predict the impact new labels will have on the performance of the model overall. Put colloquially: “how much do I think this label will help?”

There are a few flavors of expected impact strategies that vary in subtle but important ways. The first is called **expected model change**. As the name implies, here, you’re identifying which labels would lead to the largest positive change in the model’s performance. The **expected error reduction** strategy looks to measure not how much the model will change but by how much its generalization error is likely to be reduced. This (like any expected impact methodology) is actually quite expensive from a computational perspective, but if you’re trying to minimize errors (such as with medical imagery for disease diagnosis), it can be a powerful strategy.

A big difference from uncertainty sampling or QBC is that expected impact strategies look at the data space as a whole, as opposed to individual unlabeled instances. This gives expected impact an interesting advantage in active learning scenarios in that it is far less prone to selecting outliers.





# Density-weighted methods

Lastly, we'll discuss density-weighted methodologies. Instead of focusing on instances that have a high level of uncertainty (whether it's QBC disagreement or unlabeled instances the model can't make a confident prediction on), density-weighted methods consider the data population as whole.

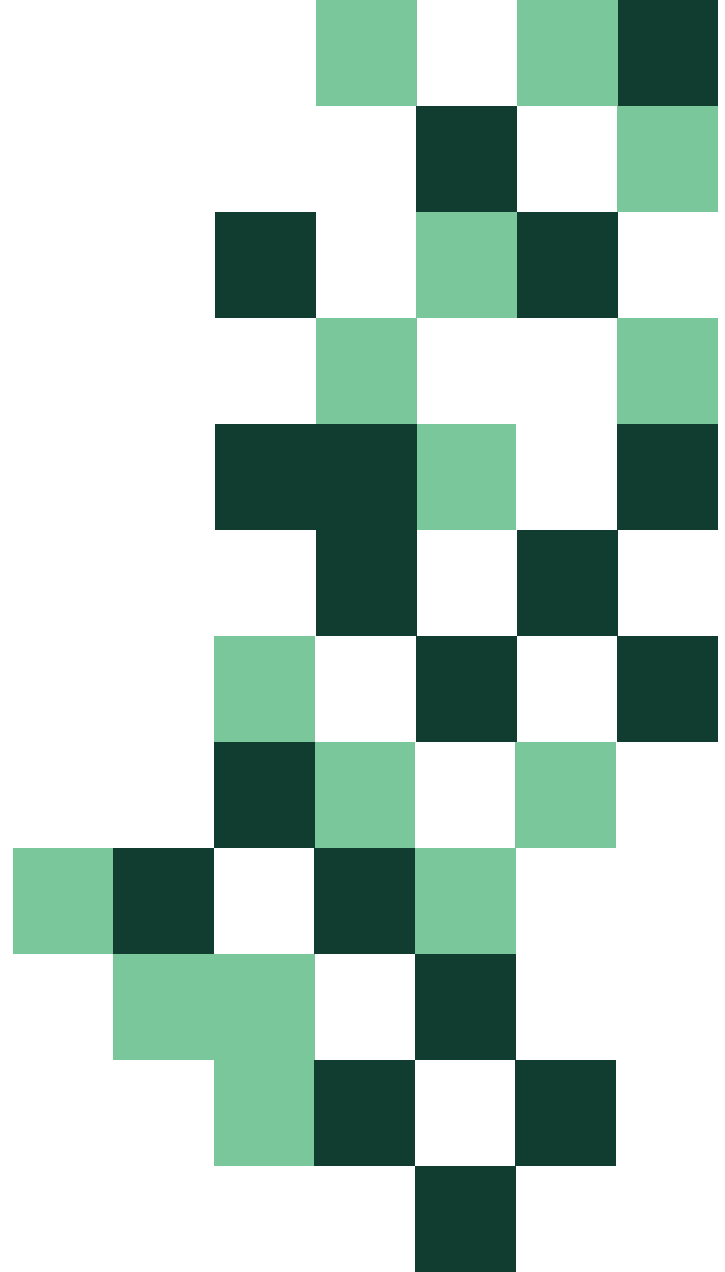
With this strategy, you're aiming to identify the data that's the most representative of the underlying distribution of the data at large. The active learner here will find data in an unlabeled set that's most characteristic of the entire set and make sure it understands those. This brings the benefit of great accuracy for the largest portions of your dataset, though it can sacrifice prediction strength on outliers or similar classes that aren't heavily represented. That said, this is a great example of where you'd use a human-in-the-loop approach to constantly iterate on your model: let the machine hone in on understanding the largest data population well while humans can handle the rows about which your model might be a bit unclear about overall.



All of these methods share certain fundamental components of the active learning strategy. All models are first trained on labeled data—generally human labeled data, at that—and all are given the ability to actively request annotations on the remainder of the as-yet-unlabeled data. The strategies they use are different but the goal is same: to leverage human oracles to better understand an entire dataset. And because they choose how they learn, active learners reduce labeling redundancy, which of course reduces both the time and money you spend making a model production ready.

Another good thing to remember is that these strategies can augment each other. Combining different approaches, either at the outset or as you iterate and optimize your model, can often lead to the most discerning, accurate model.

Of course, the next big question is a simple one: what are the best use cases for active learning? Is it right for your project? We'll try to tackle those below.





IS

ACTIVE

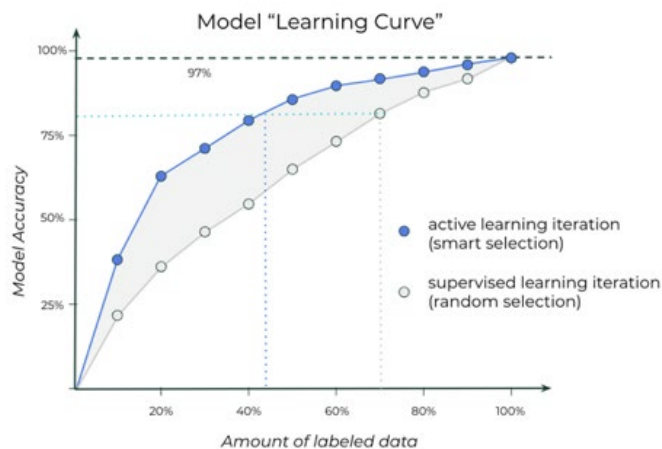
LEARNING

RIGHT FOR

YOU?



Remember the learning curve graph we used earlier?  
It's a good jumping off point, so let's refresh our memories:



What we want to underline here is that active learning strategies are much more valuable earlier in your model building process than later on. Which makes sense. After all, one of the biggest benefits of active learning is that it reduces costs and the overall amount of labels you need. There are diminishing returns after a certain amount of cycles on your model and, if your model's already performing well, active learning can certainly help, but not nearly as much as it would in the nascent stages.

With that aside, let's start by looking at when active learning is a good choice.



### Does your model require a lot of training data to reach the right level of accuracy?

If it does, active learning is likely a good choice. Why? Because active learners can actually reduce that level of training data by finding key data rows to label instead of a random sampling. Active learners consistently perform better with less training data than non-active learners. This can effectively reduce your labeling costs and the time it takes to make your model production-ready.

### Is your model prone to underfitting? Is it too simplistic?

If it is, active learning is a good choice. Active learners can query instances that will solve for underfitting, essentially finding the facets and features that distinguish squishy classes from each other. They also reduce the overall amount of training data that's needed to achieve the accuracy you're after, which can have a positive effect here as well.

### Do you have too much data but aren't sure which rows are the most informative?

If so, active learning is a good choice. You can think of it this way: if you have a vast data space but don't know precisely what's in there, random sampling poses some problems. You can easily miss important classes (or label too few of them) and large datasets, especially ones with lots of classes, require copious annotations to reach acceptable levels of accuracy. After all, the active learner will find the data rows that are most informative. It's one of its fundamental characteristics.

### Does your data have a lot of known duplicates?

If so, active learning is a good choice. That's because an active learner is generally very good at *not* requesting labels for duplicate data and, as we all know, labeling duplicative data is generally a waste of resources that can be better spent elsewhere.

### Are you running out of labeling budget but need more labels?

If so, active learning is a good choice. A central promise of this approach is that you'll find the *right* rows to label, not simply hope you'll do so. Active learning can significantly reduce labeling costs overall, so when you're running low on budget or time for that exact issues, it's worth giving it a try.



### Are your labels particularly expensive?

If so, active learning is a good choice. Think about something like pixel-level semantic segmentation for medical imagery. Those are difficult labels and, though careful instructions and quality controls can often mean your annotations are essentially indistinguishable from those of an expert, pixel labeling is quite cost-intensive.

Expensive-to-label data is especially well suited for active learning. That's because increasing accuracy with active learners requires less data than typical supervised machine learning processes. This is important from a budgetary perspective: needing less expensive labels to reach a high accuracy means spending less to get a production-ready model.

If you answered “yes” to any of the questions above, consider an active learning approach to your problem. You can get the benefit of human labeling (something we can certainly help with if you're in need) and machine learning without necessarily needing copious labels on your historical data.





WHEN

 ACTIVE

 LEARNING

MIGHT NOT BE

THE RIGHT

CHOICE 





### Are you using a pre-trained model?

Pre-trained models don't require a ton of additional data. Generally, a fine-tuning or transfer learning approach is appropriate here.

As a simple example, say you're working on an image classifier. A pre-trained classifier is going to understand edges and pixel groupings and the other various call signs of different image classes. With transfer learning, you can essentially retrain the last layer of a typical deep learning model to understand your classes, even if you're retraining a general classifier to identify melanoma.

### Does your model require a small amount of data?

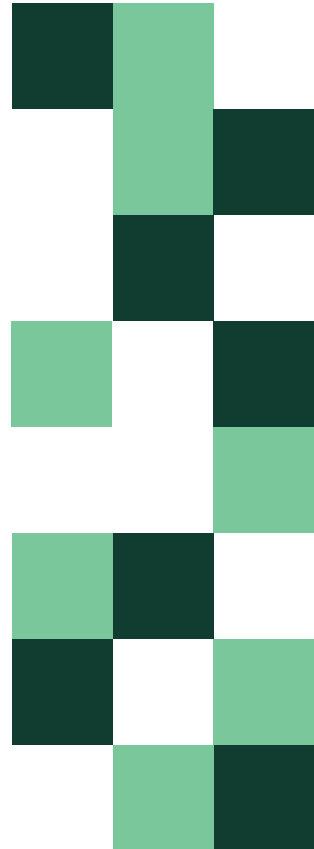
If so, active learning might not be right. Remember, active learning is ideal for finding key data instances to learn from in large unlabeled spaces. If your model is already performing well or you don't feel you'll need a lot of it to create a production-ready classifier, generally active learning isn't the right approach here.

### Does your data have tons of features/columns?

Problems with myriad classes can be tricky in an active learning context as the model will often need to see a lot of data to be successful in the first place. It may end up requesting labels on a vast majority of your unlabeled space, which is something you can handle with a typical supervised learning process or brute-force labeling schema. You may hear this called the "curse of dimensionality" and that applies here: too many dimensions to a dataset can make model training difficult generally and active learning is certainly no exception.

### Is your model prone to overfitting?

Again, this is an instance in which active learning doesn't do particularly well. Why? Because the model may actually look into the unlabeled space and select unwise instances to label, assuming the data it's overfitting for is actually correct.





ARE THERE

PARTICULAR

DATA TYPES

THAT WORK

BEST FOR

 ACTIVE LEARNING?



Active learning can work for any application. NLP, computer vision, speech-to-text, video, you name it: there's no particular data type you need to be working with. What's far more important is the section above and the answers you gave to those questions.

That said, there's no particular framework that offers a generalized approach to active learning. The strategies you use to sample data and how you empower your active learner to select unlabeled instances are the most important levers for success and, if you need help with that, our machine learning team can help set that up for you. In fact, we'd be happy to!

In fact, the hardest part of this approach is the hyper-parameter tuning aspect (i.e. figuring out the "right" pooling size or the "right" querying strategy). It takes a bit of trial and error and, yes, it's different based on use case, complexity, and the accuracy level you're aiming for. But again, we'd be happy to walk you through it or set up a solution for you.





# SOME

## FINAL THOUGHTS

Building a training set and a model concurrently is where our machine learning is going. That's what active learning offers. And while technically it's "just" a smart sampling strategy at the end of the day, active learning is adaptive and nimble. A good active learner will request labels on different rows as it learns and understand your domain space in less time than random sampling strategies that are, well, random. It will change as it goes through cycles, finding new areas of confusion and request the labels that will alleviate that confusion, eliminate redundant labeling costs.

It's of course key to have properly labeled data during both the training phase and the iterative phases of your model construction. We've been providing the best human-generated labels in the industry for more than a decade and have the tooling and expertise to make your models a success. We pioneered human-in-the-loop at enterprise-scale and, no matter your data type, your project scope, or your end goal, we're confident we can help you succeed.

If you take nothing else away from this piece, remember this:

**The fact remains that active learners get better accuracy with fewer rows than generic supervised approaches. And that's never a bad thing. Especially when it frees up a little budget for the R&D project you've been waiting to try.**

**Happy modeling!**






Figure Eight is the essential human-in-the-loop AI platform for data science teams. Figure Eight helps customers generate high quality customized training data for their machine learning initiatives, or automate a business process with easy-to-deploy models and integrated human-in-the-loop workflows. The Figure Eight software platform supports a wide range of use cases including self-driving cars, intelligent personal assistants, medical image labeling, content categorization, customer support ticket classification, social data insight, CRM data enrichment, product categorization, and search relevance.

Headquartered in San Francisco and backed by Canvas Ventures, Trinity Ventures, and Microsoft Ventures, Figure Eight serves data science teams at Fortune 500 and fast-growing data-driven organizations across a wide variety of industries.

[figure-eight.com](http://figure-eight.com)