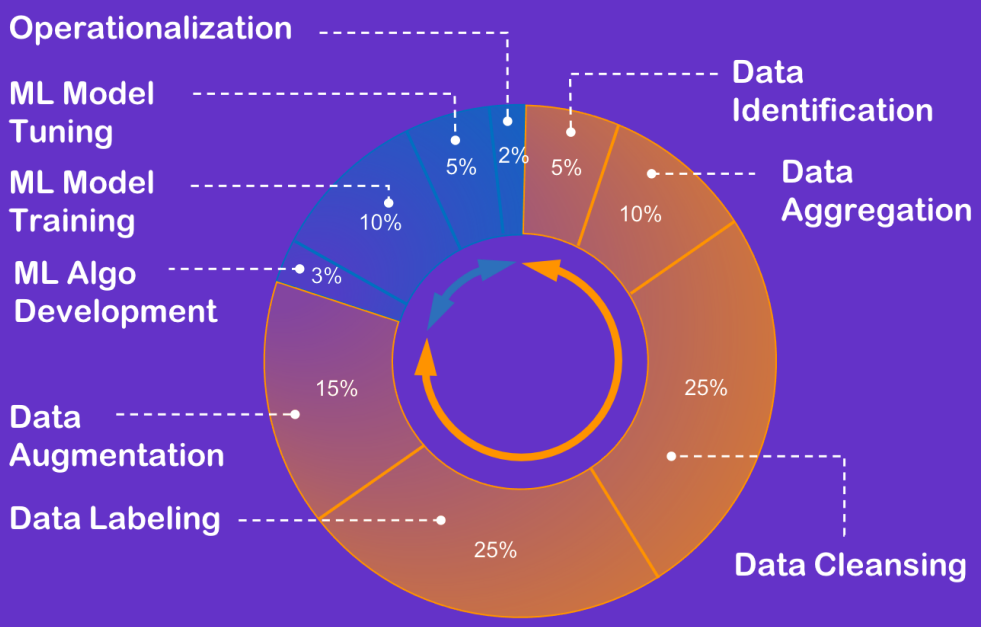
AN INTRO TO DATA PREP OPS:

THE MISSING PIECE OF THE MACHINE LEARNING LIFECYCLE

WWW.ALECTIO.COM

INTRODUCTION: WHAT IS DATA PREP?

Think about your last successful machine learning project. But instead of thinking about training and tuning your model, think about the data you used. Think about what had to happen to the raw data you had before you could actually start training that model. Which took more effort? Which took more time? Because, if your project was a typical one, preparing your data took about **three times as long as training:**



The process in orange is something we refer to as data preparation. But when most people hear the term data preparation, they think of one thing: data labeling. In fact, you'll hear some data labeling companies call themselves data prep companies. But labeling is only one of the many steps in the entire data prep cycle.

At the highest level, data prep is simply taking raw data and doing something to it so that it can be used for some sort of purpose (in this whitepaper, we're assuming that purpose is machine learning). It encompasses everything from data organization to data labeling to reformatting to ETL (extract, transform, load). Data prep is what allows us to take raw data and transform it into something we can train models with.

Data Prep Ops simply refers to the operations necessary to transform that data. That means everything from enhancements (like labeling and augmentation) to transformation (like ETL and fusion) to organization (like cataloging and curation).

Think of it like this: data is your raw ingredients. But to cook with those ingredients and actually make a recipe (a model), you're going to need to do prep work. You can't just serve a big head of lettuce with a full tomato on the side and call it a salad, after all.

That said, you could also add acquisition to the definition of Data Prep Ops. To continue our analogy, that means you start "cooking" when not just when you go to the grocery store or when you enter the kitchen but when the farmer plants a seed. That means saying data collection, generation, and even data purchasing are part of data ops.

For our purposes, we're going to include that in the data prep ops lifecycle. And we do want to stress "cycle" here. Because, of course, models aren't trained in one fell swoop. In fact, a lot of data generation and purchasing happens because existing, on-hand data isn't sufficient for our models for a wide variety of reasons. We'll start there.

DATA ACQUISITION

At the risk of belaboring the obvious: you can't do machine learning without data. For some projects, your team might have the data you need to start but need additional data when your model isn't performing as well as you'd like and you need to retrain it. Others may require you to go out and get data to get started, either by scraping or purchasing it. But chances are at some point in the lifecycle of your project, you'll need more data than what you have today.

Here are the four broad categories of data acquisition we'll be discussing:

- **Data collection**: Simply going out and collecting data. For example, if you're building an autonomous driving model, you send your fleet out to collect new video, LIDAR, and image data.
- **Data scavenging**: This involves everything from web-scraping to open source datasets.
- **Data generation**: Here, we're chiefly referring to synthetic data generation efforts that are usually more common in computer vision use cases and robotics.
- **Data purchasing**: This one pretty much speaks for itself.

Let's start with the most common and obvious way to get data: collecting it.

DATA COLLECTION

What do you think of first when we say "**data collection**"? If you're like most people, you think of the example we used in the bullet points above. That means something like driving a car fitted with LIDAR and video equipment, snapping images with a drone, gathering audio from a voice-activated device like Alexa, capturing search queries as they come in, etc. It's important to note here that data collection is **your unique data**. Other acquisition strategies like data scraping or scavenging and purchasing might get you great data from elsewhere but data collection is something you and your organization do for specific purposes. And that's actually quite powerful.

See, as you train your models, you're going to notice there are areas where it's less strong than others. In the example we used to start this section, we talked about a simple object classification model and that's a good place to start. If your model has 10,000 cat images and 100 dog images, chances are, it's going to be a lot more confident predicting cats than dogs. Smart data collection requires you to leverage that information. Instead of just scraping the web for random images or plugging in the COCO dataset, you can just collect more dog pictures.

And while "finding pictures of cute dogs" isn't exactly a challenge or a giant operational cost, collecting other types of data really can be. Think back to autonomous vehicles, for example. There, collecting data is rather costly. There are expenses not just associated with data storage, but with gasoline, with hiring drivers, with keeping cars and sensors in peak condition, with data transfer and a whole host of other considerations. Knowing that you don't need any more highway data, for example, means that your drivers won't be languishing in rush hour collecting hours of video that won't substantively improve your models.

Often, data collection efforts account for the lion's share of data we need for our models. Being **smart** about how we collect it has a cascading effect for every other step in the data prep lifecycle, not to mention the accuracy and success of your models generally. To do this as intelligently and leanly as possible requires you to understand what data your model needs to see next and what data it doesn't. And Alectio's core technology can really help with this. We can identify the data that your model is hungry for at every step of its training process so you're getting the best use out of every label and piece of data you collect.

SYNTHETIC DATA GENERATION

Another way to acquire data (one that's really growing in popularity, by the way) is by simply creating it.

Synthetic data generation is probably most closely associated with facial recognition, object recognition, autonomous driving, and robotics—in other words, images are the most commonly generated data type. You do see some synthetic data in NLP and a bit more in biotech, but, again, it's far more common to create images than anything else, largely because of the tremendous amount of research done on generative adversarial networks (GANs). And as a testament to its growing efficacy, there are companies who specialize in creating this data for machine learning applications—Unity, Synthesis AI, Tonic.ai, Hazy, and AI.Reverie come to mind, but there are many others as well.

But there are real challenges with creating synthetic data. For starters, you need a lot of real data upon which to base your synthetic data. You can't simply instruct a machine to create the faces you might see on thispersondoesnotexist.com. The models there have seen tons of examples before they started churning out realistic facial images. That means that if you're working with novel or fairly rare data types, synthesizing data is going to be a lot harder and likely something a third party will have a harder time helping with.

In fact, here's what we're talking about. On the left, you have a great example of synthetic data. In the middle, everything looks fine until you notice the strange artifact on her forehead. You'll see smaller oddities in a lot of synthetic data like background distortion or women with two different earrings. And then, there's the image from thishorsedoesnotexist.com on the right. Whatever it is, it is decidedly *not* a horse.

Additionally, synthetic data also requires a ton of compute resources to create, so much so that sometimes, simply collecting more data can be more efficient from a budgetary standpoint, especially if your synthetic generation is churning out stuff like the images in the center and the right above.

One additional challenge is we're still in the fairly early stages with research around synthetic data. In fact, sometimes the generative algorithm just gets things wrong. We don't fully know its impact and, regardless of how great it looks, there can be small, sometimes imperceptible issues with the data that gets created. For example, say you wanted to create snow data for your driving algorithm. It seems fairly easy to create images or even just augment ones you already have to make them "snowy." But are you getting the nuances? The way snow melts on highways or how it interacts with tires and cars in motion? It might seem like you have a great synthetic image but the small differences can teach your model the wrong things. Synthetic data can unfortunately be destructive to model accuracy.

That said, there are domains that are well-suited for it and there are times when simply collecting more of a particularly hard to find kind of data can be onerous. It's likely **best used** for common use cases that don't require unique, boutique data and use cases that can tolerate less accurate outputs.

DATA SCAVENGING

The third way to acquire data is tried and true and something many of us used in school. That's data scavenging. For our purposes here, scavenging refers mainly to datasets you didn't collect, purchase, or create from whole cloth. We're mainly concerned with **web scraping** and **open source data**.

Let's talk about open source data first. Open source data can come from places like the UCI library, canonical datasets like COCO, Kaggle competitions, research papers, finding one on Google's dataset search page (datasetsearch.research.google.com), and so on. A lot of this data, especially the ones that get used in research and academia are both clean and well-labeled. And that can be great for certain use cases but it doesn't exactly mirror how data is in the real world. Training an object recognition model on clear, in-focus images seems great at first but that model might fall apart when it sees messy, real-world pictures.

Generally, this is a reason why few organizations base their models solely on open source data. This data can be great to buttress your models and to bootstrap them at the beginning of your training cycles. In fact, you might rely on such open source datasets without even realizing it whenever you use a pre-trained model, or when you use transfer learning. Unfortunately, open source data isn't often representative of real world data.

We likely don't need to say a ton about web scraping, but it's a solid way to acquire data for certain use cases. NLP use cases, for example, can benefit from real world social data that can be scraped from Twitter or Yelp or any other site. But it's a lot easier to scrape this kind of data than it is to collect or synthesize it.

There is another important thing we'd like to talk about with data scraping or scavenging: for certain kinds of data, **it can be the best and sometimes only ethical way to collect additional information**. For example, think about what happens if you need more video of car accidents for your autonomous driving models. You can't very well head out to Main St. and run into some cars for the sake of your algorithms.

DATA PURCHASING

Lastly, we want to spend a few paragraphs talking through simply purchasing data. The example we just gave about scraping tweets can be good for some use cases, but Twitter understands this data is valuable so they restrict the amount of data you can freely pull from their site, even with developer credentials. For full access to this data, you need to purchase their so-called "firehose." And if you're creating models to help target ads based on social media behavior, that is likely going to be a really smart move.

That's of course just one example of data purchase though. Organizations like Meltwater or Nielsen sell data. Some companies even package open source data and organize it to save ML practitioners the time and effort it takes to do so on their own.

But there's a real challenge here. Say you purchase autonomous driving or robotics data. Are your cameras the same as the ones used in the data you just bought? Are the sensors mounted in the same way?

If not, you're likely injecting some pervasive biases into your model. Since companies often purchase large amounts of data, you want to try and account for this at the outset. And it's worth thinking about this sooner rather than later. After all, it's likely there will be more and more companies selling data in the coming decades, and not just companies for which selling data is their main business. Just as an example, think about a company flying drones over urban areas to forecast traffic patterns. They're also getting data that might be useful to insurance companies or weather services—why wouldn't they sell some of that data?

HOW ALECTIO CAN HELP

As we mentioned above, acquiring data isn't something you just do up front. Many times, we acquire data to fill in gaps in our models, to show it additional examples of classes it's hungry for, or even just to test our model on real world information.

But acquiring data shouldn't be done haphazardly. You need a strategy so you don't simply feed your algorithms more examples of things they already know or data that might inject harmful biases. Alectio's core technology identifies the specific classes and types of data your model needs in every subsequent training cycle so you can acquire data intelligently.

DATA ENHANCEMENT

Think of machine learning like baking bread. In this analogy, data is your flour: it's the essential ingredient to making a nice loaf. But you likely aren't growing and milling your own grain. To continue the analogy here, data prep is the process of doing just that: turning a grain into flour you can use. And if data acquisition was harvesting your wheat, enhancement is separating the wheat from the chaff. You wouldn't throw whole stocks into your mill, after all.

When most people think of data enhancement as a category, chances are the first think they think about is **data labeling**.

Now, chances are, most readers are pretty aware of what data labeling is, but if not, it's just the process of applying tags or labels to a piece of raw data so you can train a machine learning model with it. On images, that could be anything from simply identifying an image ("this is a picture of a cat") to something more complicated like drawing a bounding box or labeling individual pixels ("there is a cat inside this box" or "these pixels are a single cat"). Audio can be transcribed or you can apply labels to sections with white noise or areas of interest. Text use cases can be anything from identifying proper nouns (usually called "named entity extraction") to summarizing entire paragraphs. And that's just a start,

In the end, what you're left with is a piece of data that a human or machine applied some sort of label to. So instead of an image being just a collection of pixels, a machine can now understand what's actually in those pixels.

We'll also be covering **data augmentation** in this section (i.e. flipping an image, injecting noise, etc.), which is a common way to enhance data you already have and grow your dataset. But let's start with something we're intimately familiar with here at Alectio: labeling.

DATA LABELING

On its face, data labeling is deceptively simple. Most people think it just means sending their data to some group of people, usually external, and those people provide the labels needed for machine learning. And to a certain extent, that's pretty accurate.

But like most things, a simple definition just scratches the surface of what data labeling is. There's a lot of nuance and complexity here. The choices you make about **who** should label your data, **what** they actually label, and **how** they do it are crucially important for the success of your models.

WHO LABELS?

When we defined data labeling above as "the process of applying tags or labels to a piece of raw data so you can train a machine learning model with it," you'll notice we didn't mention who actually applies those labels. That's an important point and one we want to talk about briefly here.

For starters, let's just focus on human labelers, namely the two major types of external labeling providers: the **crowdsourced** model and the **business process outsourcing (BPO)** model.

Crowdsourced models are generally a platform where people from around the world can label data in their spare time. Some of these people make a living of it but many others spend a few hours here and there reading instructions then labeling data. Crowdsourced models can push through a ton of data rows because the barrier for entry is lower than with a BPO but they're often a little less accurate because there's no central organization controlling their output. They're best for jobs with a lot of data and a lot of easier or subjective labels. Jobs like object classification ("is there a cat in this picture?") or sentiment analysis ("is this tweet positive or negative?") are great examples.

BPOs are organized differently. Instead of people logging in from their machines at home and providing labels, BPOs are more like dedicated labeling shops. Here, people work as data labelers first and foremost. They're trained, they have more experience, they have project managers and colleagues to help them and keep them accountable for accuracy. Some BPOs have expertise and clearances too, like specializing in cellular semantic segmentation or having HIPAA or government clearance. Essentially, BPOs produce higher quality labels but they can't push through the volume of crowdsourced providers.

There is, however, another dichotomy to consider and that's whether or how much of your data should be human-labeled in the first place.

There are a few different ways to do autolabeling or ML-driven data labeling. You can leverage pre-trained models, but that usually works best for more common use cases. Think using an object recognition model trained on COCO for a similar model you're building in house. We offer an auto-labeling module at Alectio in fact. You can also use a Snorkel-style ensemble labeling approach where humans create rules but don't do the row-by-row labeling. But the issue is largely that autolabeling works really well for some use cases but not unique ones---in fact, it's a similar issue to the one we discussed around synthetic data.

That said, you'll often find people relying on a combination of both human and machine labeling. This is called **human-in-the-loop machine learning**. Typically, a model looks at unlabeled data, makes a prediction about what the piece of data is and then, if it's under a certain confidence level score or another relevant metric, will send that data to a human labeler.

Essentially, the idea is to farm out only the hard stuff to people and let the machine handle the simple labels. This is a great way to reduce costs without sacrificing accuracy and it's something we offer in our Alectio Data Labeling Marketplace experience. Essentially, our marketplace is a collection of some of the best BPOs on the planet ready to label your data on a project-by-project basis. If you need experts in LIDAR one week and segmentation the next, we can find that for you. BPOs get a few minutes to take your job so you aren't waiting around working on their schedule. And if our autolabeling solution can help, we'll tell you that too, showing you our predictive accuracy on classes our autolabeler understands.

You can read a whole lot more about that here (alectio.com/labeling-marketplace) but even if you don't use Alectio, a human-in-the-loop method (like the one Dataloop employs) is a great way to marry the benefits of machine and human labeling for great accuracy without the onerous costs of human-only labeling.

WHAT GETS LABELED?

Now, let's talk about **what** you label. And no, we're not talking about the data type. We're talking about how much of your data you label and how you choose that subset. It's about labeling the right data.

See, one of the biggest mistakes many ML teams make is over-labeling. They get labels on classes their models already understand. They prioritize the wrong examples. They try to fix underperforming models with not just more labels, but more of every label.

Much like coming up with a data collection strategy like we outlined in our first section, you also want a labeling strategy. Your models don't need every single piece of data labeled the same way by the same amount of people (or machines) and you don't want to just keep adding more and more and more. It's costly and, frankly, it's counterproductive. Rather, consider leveraging something like active learning to understand what data should be labeled. Be strategic. Don't simply label everything. It's costly and often doesn't improve model performance!

HOW DOES IT GET LABELED?

Lastly, no two providers will give you identical labels and no two companies have identical data labeling needs either. That means coming up with a unique strategy that works for you and the labeling provider(s) you choose.

Understand their tools and know how they'll interact with data you already have. Spend time writing exacting instructions that account for edge cases and nuances so the labels you receive are the ones you need. We've seen many machine learning teams skimp on the time and effort on this crucial step—connecting with your labelers and making sure you're understood—and the results can be really detrimental. We get into a lot of detail about instruction writing in our **5 ways to save on data labeling** white paper (available on our website), but understand that there's almost always going to be some back-and-forth with your provider as you figure out exactly how to describe the work you need done.

Remind yourself that these labels will power your model. Remember the old garbage in, garbage out (GIGO) adage. Then take a few extra minutes to talk with your provider and get their advice about how best to get the labels that will make your project a success.

5 QUICK TIPS FOR LABELING INSTRUCTIONS:

- **SHOW EXAMPLES**: You're intimately familiar with your data. Labelers won't be. Show examples of your classes, how you want bounding boxes placed, or any relevant information you feel is important. Show, don't tell.
- **ALLOW FOR N/A**: Sometimes, a data row might be corrupted or not have any of the classes you care about. Letting labelers mark "n/a" means you won't be forcing them to give you bad labels.
- **GET FEEDBACK**: Your labeling partner is the expert. Make sure your instructions are adequate by asking them if they are. If they won't help, you should look elsewhere.
- **EXPLAIN EDGE CASES**: People on the street are different from people on a billboard. But labelers won't know unless you're explicit.
- **LABEL YOUR OWN DATA**: You'll uncover a lot of those edge cases and other issues you might not have considered if you label your own data for just thirty minutes or an hour. Trust us: it's worth your time.

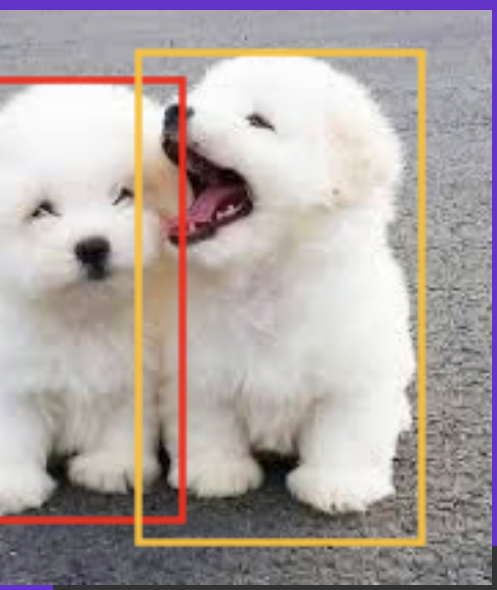
LABEL VALIDATION

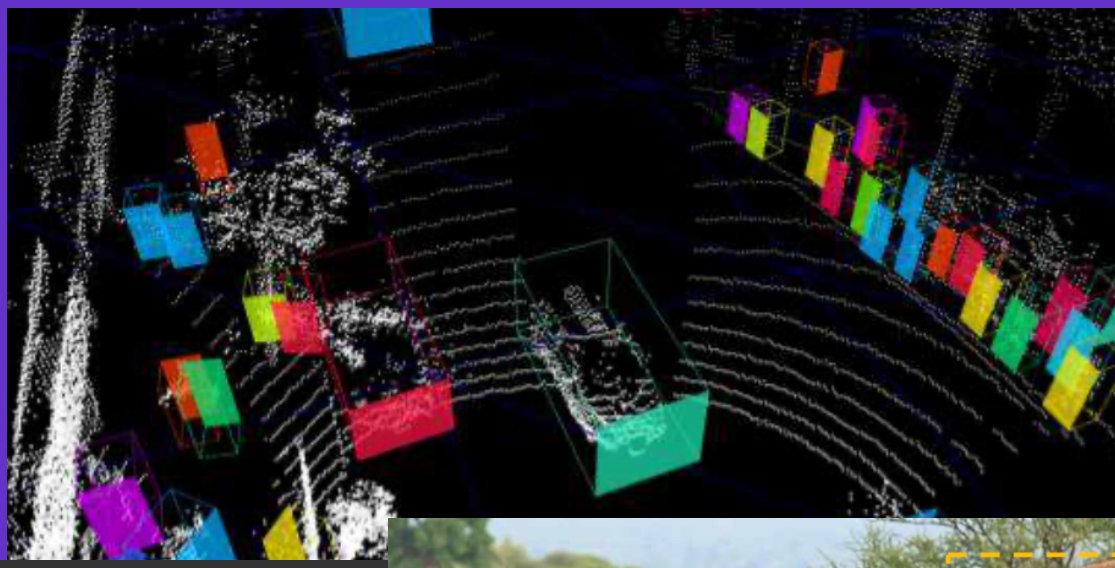
One final and important part of the data labeling process is actually **validating** the labels you receive. In essence: did you get what you asked for? Are these labels safe to feed your model?

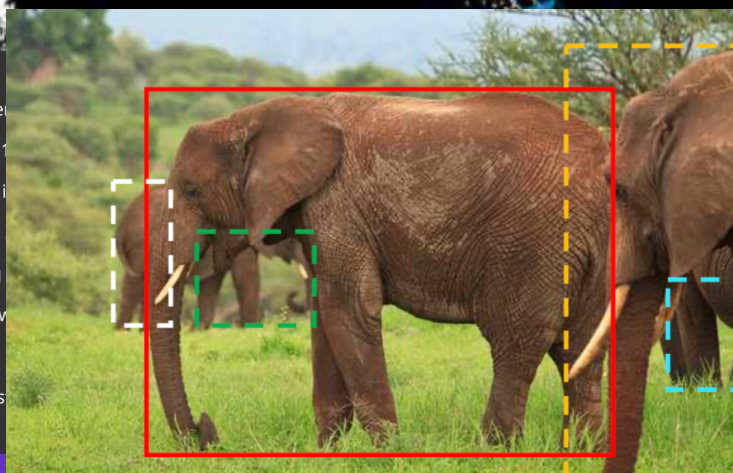
First and foremost, no matter which labeling provider you use, they're likely doing their level best to give you good labels. Bad quality is the surest way for them to lose business, after all, and they have no interest in giving you garbage when there's hefty competition in the industry already.

There are some of the few common ways to find errors in your labels or to avoid those errors from occurring in the first place. One way can be a simple test of each prospective labeler to make certain they understand your instructions. That means labeling a handful of your own examples and programmatically checking their success against your own labels. Labeling companies or individuals who don't reach that threshold should work on a different task instead of yours. Easy enough, right?

That said, even the best labelers do make errors from time to time. Even ones that understand your instructions well will mess up and have a good faith mistake. That's why some companies use multiple labelers (or redundancy) to guard against this. One labeler mistaking one class for another can be overridden by two or three that get it right.







Sia Kate Isobelle Furler (/ˈsiə/ SEE-ə; born 18 December 1975) is an Australian singer, songwrite
music video director.[1] She started her career as a singer in the acid jazz band Crisp in the mid-
in Adelaide. In 1997, when Crisp disbanded, she released her debut studio album titled OnlySee
Australia. She moved to London, England, and provided lead vocals for the British duo Zero 7. In
Sia released her second studio album, Healing Is Difficult, on the Columbia label the following
and her third studio album, Colour the Small One, in 2004, but all of these struggled to connect
mainstream audience.
Sia relocated to New York City in 2005 and toured in the United States. Her fourth and fifth s

There's also pre-training and training-time validation. Validating the quality of the labels is actually no small feast for companies; in fact, even among those who rely on third party data annotation companies, many have internal labeling validation processes to make sure that their data is indeed 'safe' to use. This can manifest itself in simple forms, with the ML team pitching in when they notice defects in their training data, but it sometimes is more formal, with an entire team of internal data validators hired full time to make sure the labeling accuracy promised by the labeling company is reached.

There's no question that validating labels can be as expensive and difficult as data labeling itself. That's why, in most cases, companies prefer relying on redundant labels as opposed to validating them at the end. This means that for each record, several annotators are asked to provide an annotation, and all these annotations are combined. This reduces the probability that a mistake will sneak in the dataset, making validation less critical.

But redundancy is not always realistic, and besides, it isn't cheap. In that case, there are two options to capture and correct mistakes:

1. Have a process (involving humans and/or machines) to confirm the labels; this needs to happen after the labels are collected and before they are used

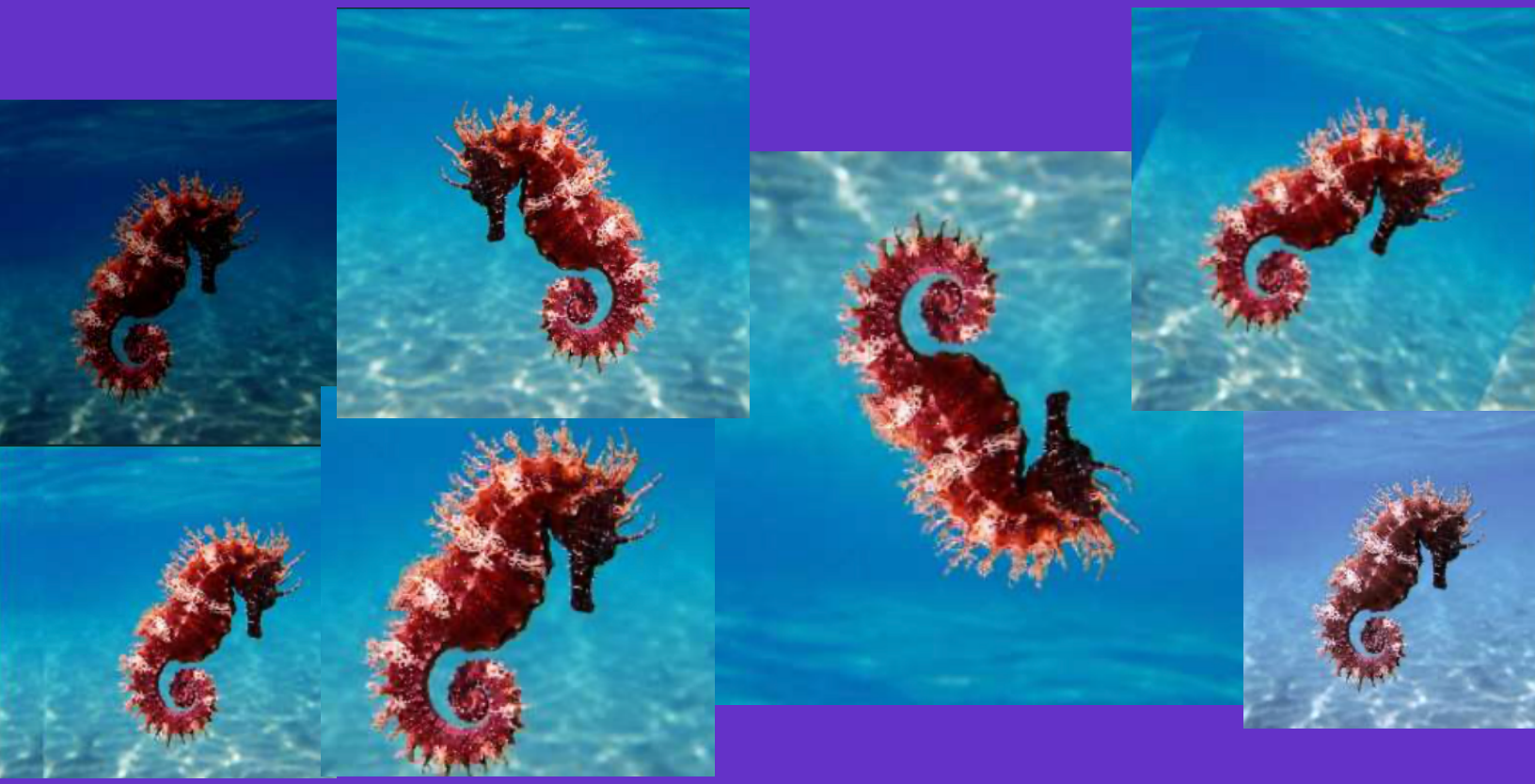
2. Use the labels as-is, and use the ML model that will be trained with that data identify weaknesses, potential mistakes, and automatically send the most contentious records to be relabeled. (in-training validation)

Additionally, if you use our labeling service, we provide a ranking algorithm that sorts data by order of contentiousness which shows you the records that are likelier to be mislabeled. That isn't to say they are mislabeled, mind you, but that we've found signs and meta-signals that they could be. It's worth spot checking labels in those records to make certain everything looks good. If they're on the level, your other labels likely are as well.

DATA AUGMENTATION

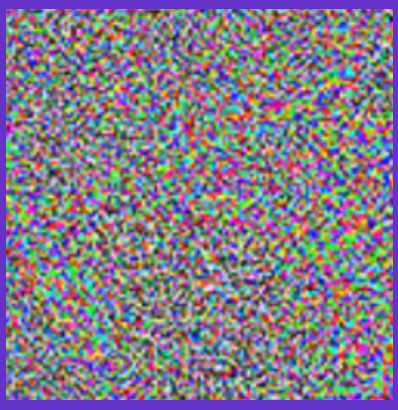
The last part of the data enhancement step of the data prep lifecycle is augmentation. Augmentation is a great way to shore up smaller datasets, taking a single example (say of a sea horse) and creating dozens of additional data rows of that same example (ever more sea horses, in fact).

The simplest augmentations are things like taking an image (you'll never guess which sea animal we'll be using for our example) and flipping, rotating, re-coloring it, zooming in, etc. to create more examples for your model to learn from.



Here, we've effectively created **seven additional versions of the same sea horse** for our model will learn. If you've done augmentation before, you know how a simple rotation can lead to a different predictive accuracy, but these kinds of rudimentary image alterations aren't the only kind of augmentation we're going to talk about here.

There are other types of augmentations that are more subtle than traditional geometry-based augmentations, like the **injection of random or gaussian noise.**



By injecting a light amount of noise into the image on the left, an amount that's basically imperceptible to most of us, and suddenly the machine changes its prediction from "that might be a panda" to "that's definitely a gibbon."

These augmentations can help gird your model against later bad predictions by correcting it and giving it a better understanding of what the core attributes of each class are.

There are also AI-driven augmentations powered by generative adversarial networks (GANs), like changing the angle of a face for facial recognition or generating snow on images to simulate winter conditions.

Above, we have our original image on the left and a style image of snow in the middle. A GAN is used for "style transfer" to create the picture on the right. In some ways, it's a pretty remarkable transformation, but there are real issues with what we've created here (namely the river and the colors of the homes). You need to be careful when you use this strategy.

The fundamental idea of all these augmentations is the same: it's to give your model additional, different examples that give it new information upon which to make its predictions. But there are a few things to keep in mind if you want to do augmentations for yourself.

First off, you don't want to simply create a whole host of augmentations. Remember: these aren't "real" images. Earlier we gave an example of creating synthetic data for a snowy day on the road and all the problems that could arise from that. Take a look at these **two real, non-augmented images below**:

It's going to be exceedingly difficult to recreate the snow spreading out in front of those plows on the image to the right or even just the lines of asphalt the one on the left. In fact, you can't get an image like this through augmentation. The point here is that some data is impossible to create and neither synthesizing data whole cloth or augmenting it makes it "real."

There's also the fact that there hasn't really been sufficient research concerning augmentation and how all this pseudo-redundancy impacts training. It's hard to know if models are picking up bad habits or identifying the wrong types of features from augmented data that could harm them when they see real world examples. And of course, sometimes, more and more data isn't the answer to the issues your models actually have right now.

We'd recommend being fairly deliberate when using augmentations and coming up with a strategy that fits your exact needs. **There's simply not a one-size-fits-all solution for how to augment your data and you'll need to experiment to get there.** Take those images above as an example, the pictures of those cars on a snowy highway. Augmenting them by flipping them horizontally might be great, but vertically? That's likely only useful if you're planning on driving around the International Space Station. And probably not then either.

Chances are, this is not a valuable augmentation for your model

DATA TRANSFORMATION

At this point, we've somehow acquired our data and enhanced it by labeling or augmented it. What we're going to talk about next is more data engineering than data prep, but make no mistake: there's no machine learning without data transformation.

We're going to look at three types of transformation in this white paper: **formatting, feature engineering, and data fusion**. Formatting is more or less extract-transform-load (**ETL**)––marrying databases, making data formats uniform, and organizing it so it can actually be used and analyzed by machine learning scientists. This is an incredibly important part of any data project and there are plenty of companies better equipped to talk to it outside of us so we'll leave it to the experts.

But ETL does bring to mind one important point: **the fallacy of the full stack data scientist.**

When data science jobs started popping up across myriad industries, you'd often see job postings that expected people to handle essentially everything related to data. They were supposed to know how to do ETL, how to write models, how to label data, engineer features, and basically be knowledgeable about each step of the data prep and later the machine learning process. This is, of course, preposterous. You'd never see a job posting for a "marketer" who was supposed to be able to write content, throw events, design swag, head up rebrands, handle social media, editing, and web development, run demand generation campaigns, and do product marketing on the side. That's unreasonable. We all know that. But a lot of companies really *did* look at data scientists this way.

Thankfully, those days are mostly in the past. But if you're in a younger organization, it's important to remind whoever's doing the hiring, that while machine learning is the new, sexy thing, it likely will not work without a data engineer on staff. It's a role that's too often overlooked and a lot of us who have had the pleasure of working with a great data engineer realize how invaluable they really are.

With that said, we're going to leave fusion and ETL to the experts but we do want to spend a moment talking about feature engineering and how it ties in well with one of the themes we've been coming back to time and time again: more isn't always better.

FEATURE SELECTION & ENGINEERING

Chances are you're intimately familiar with feature engineering, but if not, a quick definition. Say you're creating a sentiment analysis model. The text is your raw data, but each label and attribute of that data is a feature. Those features might be something like whether the comment is in ALL CAPS or the presence and amount of exclamation points or emoji. The trick for machine learning practitioners is to figure out which features, labels, and attributes are most predictive. More exactly, it's figuring which combination or what relationships between those attributes is most predictive. Doing that is **feature extraction (or selection) and engineering**.

Moreover, what's really important to underline here is that you simply can't use every feature or every relationship between any given number of features. This is sometimes called the curse of dimensionality: the size of this problem is just too large. Feature engineering is about finding the best set of features to predict an outcome and that involves jettisoning the ones that aren't as helpful from your model (though technically, that's feature engineering). You aren't removing these from your databases, of course, but simply not using them in your models.

Now, there are a few different ways to determine the best set of features for a given model. The first we'll talk about here is leveraging **domain experts** to do so.

While machines can be great at uncovering novel relationships here, sometimes, old-fashioned expertise is the way to go. Say you were building a model to predict home values. Knowing what's in fashion today––be it sleek modern kitchens or secluded cabins sitting on unspoiled acreage––is incredibly important. You might know people are cooking more during a pandemic so folks are after bigger kitchens. You might know which neighborhoods are poised to enjoy a renaissance. You might know all manner of nuanced things about the housing market generally or in a particular area.

And while that doesn't mean you'll find the right features the first time you try, you'll be able to lean on your domain expertise to start coming up with smart ideas to test.

That said, if you can't lean on domain expertise (or even if you can), there are other ways to find more optimal features for your models. There are statistical methods designed to identify the features that correlate best with the outcome you're trying predict (such as Principal Component Analysis (PCA) or Manifold Learning).

But engineering features this way can be expensive, namely because hiring experts who understand this can be rather costly. One smart trick companies are using is to re-use the features designed by experts across projects or even departments in their organization. For example, features that were useful to predict which patients were likely to come back to the hospital might be useful designing better treatments as well. That said, this doesn't work for every problem, can be rather costly in its own right, and can take your ML experts off other projects.

The good news is that there are feature stores like Michelangelo, Tecton, or the AWS Feature Store where you can purchase things that will help with **versioning, traceability, feature definition,** etc. You can also find the most predictive features mathematically, though that's an entirely different whitepaper.

The point is you don't *need* use everything. You can't use every feature and every relationship between features in your model. They'll quickly balloon in size until they're unwieldy and ineffective. You need to be smart and strategic here, just like how you should have a data collection strategy and a data labeling strategy. The era where we solved problems with more features and more data is one we should all be excited to leave behind.

DATA TRIAGING

The last step in the data prep cycle is something we're calling data triaging. Broadly, we parcel this out as data cataloging & structuring as well as data selection. We'll be spending more time digging into selection but we do want to spend a quick moment going through cataloging and structuring first.

ML experts have long shown a lot of excitement towards Big Data, which is completely normal: after all, more data usually means more diversity, and hence, a better chance to find data that's helpful to train a model. But with ever larger datasets, we should start thinking about Big Data differently, and see the opportunity that we have to leverage it not for its size, but to be pickier with the records we chose. Think of it like a bookstore: we don't love gigantic bookstores because we're going to read everything on every shelf but because we're more sure that they might have the few novels we're really excited about. After all, you're far more likely to find what you want in a three story bookstore than the little shop at the airport that sells candy bars too.

Data cataloging is part of this step but it's not necessarily one we all have to do. In fact, you'll find it most commonly practiced in companies that sell data as a service, though organizations with tons of data and multiple streams and products do a fair amount of cataloging too.

Here, think of classifying data by its relevance to a particular use case. If your company is a drone imagery company, for example, you might catalog data according to whether it's in a residential area or an urban center. You might have images of parking lots at various malls you could see to people trying to forecast buying patterns in a certain region or predict how big this year's holiday shopping season might be. You might want to have images of the rain forest cataloged for an environmental initiative your organization has volunteered to help with.

Those are just a few quick examples, but the idea is that you're cataloging and packaging data for a specific use case out of your larger set of data. You're being picky within the big data at your disposal.

Then, there's **structuring and searchability**. Here, the goal is to access the insights in your data more efficiently, allowing you to find edge cases, corrupted data, or any particular caetgory of interest. Should your model show poor performance in snowy weather, you'd automatically want to include more training examples of snowy screens - for better or for worse. Companies like Aquarium Learning and SiaSearch allow you to do just that. The problem is that you might just pass on to your ML team the ability to inject biases into their models. Why is that? Because when your model performs poorly when it snows, chances are, what you may need is a combination to snowy screens indeed, but also light vehicles and high luminosity.

A little data structuring never hurts

This can be done by data captioning, for example, but the idea is getting more explainability and searchability into large datasets you can't possibly know everything about. These are all laudable goals, certainly, but you want to keep in mind that making data searchable doesn't mean you're determining what data is actually useful to your model. Be careful about leveraging these methods too much as you and your team can inject biases based on what you assume is useful vs. what actually *is* useful to your model.

Which leads us to **data selection.** See, given any large set of data, we all know that some of that data is more useful than other data. In fact, some data can actually be actively harmful to models because it was mislabeled or corrupted somehow, maybe from a faulty sensor or a cracked lens on a mounted camera. Knowing what data your model finds most useful and what data will reduce accuracy is often overlooked but it's something that can unlock better performance, reduced training times, and significantly reduced budgets.

A great and, we think, underutilized approach for this issue is **active learning.** It's a backbone of our approach at Alectio and we've seen it bring real dividends for our partners. Essentially, instead of training your model on a random subset of your data (or all your data), you train your model incrementally, in smaller batches, paying attention to what the model reacts to and making inferences about what data should then be in the next small batch. Figuring out that next batch is something called your **querying strategy** and finding the exact right strategy for each individual project is a real challenge. A rudimentary strategy is called **least confidence** where, you guessed it, you train your model on examples it's least confident about.

Of course, if least confidence worked for every model, there wouldn't be much to talk about here. As we mentioned, finding the right mixture for a querying strategy is a lot like finding the right weights for your model or fine-tuning your hyperparameters. It's something that takes both experimentation and expertise. (And it's something we've spent a lot of time on here at Alectio. We'd love to show that to you if you're interested!).

That said, regardless of how you go about it, if your team is treating all data as equally utile, it's probably not the best idea. Remember that representativeness often doesn't equal usefulness. In fact, you may actually need more corner cases that are hardest for your model to learn. Make a concerted effort to identify what's helping your model and what's confusing it so you can prioritize training on the data your model needs most.

PARTING THOUGHTS

Even though data prep typically takes far more of a team's time and energy than machine learning, it often gets less attention, credit, and budget. But there's simply no machine learning without data prep.

Remember that every step of the data prep lifecycle is crucially important and that each step depends on the others. You can't do anything without acquiring data, you can't do most machine learning without some kind of labels or augmentations, and you can't get at your data unless it's transformed and manageable. Skimping on one is going to have deleterious effects on the others.

Data prep is so often treated as a second class citizen in ML. It shouldn't be. Preparing your data is the first step to making your ML initiative actually work in the real world. And at Alectio, we're committed to helping make your data prep easier. We can help you identify the smart data inside your big data so you don't train your model on harmful or useless information. We can help you decide which data to collect or acquire next. We can train your model with targeted batches of data that unlock model performance. We can get you in touch with labelers that are right for your current project and are available to label on your schedule. If you'd like to learn what we can do for you, just reach out. We'd love to get you started.

ABOUT ALECTIO

At Alectio, we help the most innovative companies in the world train better machine learning models with less data. Our platform employs an ensemble approach that includes active learning, reinforcement learning, meta-learning, deep learning, and more to identify what data is actually helping a model learn and what data is holding it back. Alectio uncovers the smart data inside your big data, unlocking model performance and saving machine learning experts both time and money. Visit us at alectio.com