Crawl, Walk, Run: Bringing Data Science into your Organization

Throughout my career as a data scientist, I've been lucky enough to have a few opportunities to build data science teams, processes, and models from the ground up. Introducing data science into your organization can feel overwhelming, so I've put together some recommendations to help you along the way.

While it's incredibly tempting to jump into the deep end and start training Deep Learning systems to solve the hardest problems that have been plaguing your company, I'd recommend a slower, progressive implementation based on my experience.

One way to incorporate data science into your organization is a three-phased approach that I'll refer to as Crawl, Walk, Run.

If you don't have any data science implemented yet, the Crawl section will help you get started. The Walk and Run sections will still be relevant, though, as they will help you establish your initial workflow while keeping "what's to come" in mind.

If you already have a data science group successfully developing and deploying models, some of the information in the Walk or Run sections might provide additional practices you can implement if you aren't focusing on them already – like quality control and responsible modeling.

We're crawling!

Defining a workflow

If you're just introducing data science at your organization, you get to build out the entire workflow for the first time. While it can understandably feel a bit overwhelming, it is an exciting time, and there are many free resources available online to help guide you along the way.

You get to:

- play with a lot of fun technology and investigate which tools work best for your organization,
- define best practices for data collection and usage, and
- discover how to effectively present the results to the end-users.

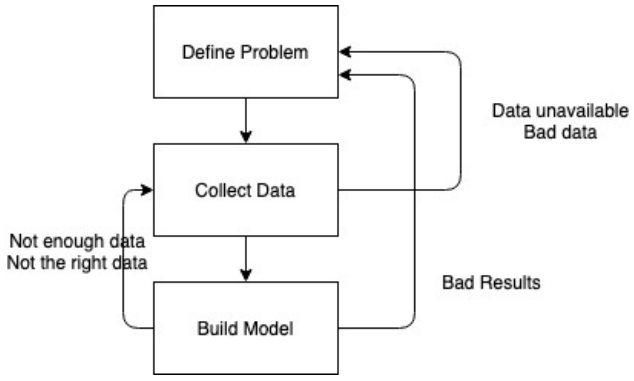
Part of the process is building organizational confidence in the models and learning how to set realistic expectations of modeling. A large part of that confidence depends on everyone understanding the data science lifecycle.

Understanding the lifecycle

There are five basic steps to a data science lifecycle:

1. problem discovery,
2. data collection and cleansing,
3. exploratory data analysis,
4. model development, and
5. communicating the results.

While these are the basic steps, it is important to remember that the lifecycle is not linear, but, rather, a feedback loop. There is a high potential you may have to start over completely or go back a step or more at any point in the process (illustration).



After attempting to build the model or even after analyzing relationships between the data, you may realize you haven't collected all the data you need. You may need a larger sample, or you may find entire fields that are important to the analysis are missing. Sometimes you may collect all the data only to find out after the initial analysis that the data you collected was bad.

Even if you can collect high-quality data that doesn't mean you'll have a model that produces valuable results. A poorly performing model can happen for many reasons. It could be that there just isn't a pattern in the data that you're analyzing–it's all noise. Or, if you're trying to assess the success of a newly implemented program, perhaps enough time hasn't passed to see measurable changes. There's also always the possibility that the original question may not be defined well enough.

Because any or all the above may occur, it's important for all involved to realize that the data science lifecycle is *always* an iterative process. (This includes both those building the models and those hoping to utilize the models.) We'll frequently make discoveries in later steps and have to go back a step or two to build better models.

Start small

While it's tempting to try to get buy-in within your organization by tackling that big problem that everybody has been wanting to solve, it's often better to start with a small challenge that has a well-defined problem statement and easily accessible data if this is your first time building models.

Building confidence in your models and figuring out your workflow is part of the process. It's good to test all phases of the process on a less-complicated model to ensure the process works without adding in complications that can come from working with a more difficult/complex model.

Defining your problem, goals, and measuring success

One of the reasons a data science task may fail is because there isn't a well-defined problem statement or goal. Knowing that you want to improve the outcome of something but not knowing exactly what constitutes an improvement can make it difficult to build a model and collect the data you need. A well-defined problem should identify what is being measured, how that measurement relates to the goal, and what counts as success.

Keep in mind that usage of a model as part of a preventative measure may change the underlying data and may make your initial forecasts look incorrect because your results have changed the behavior of the data being measured. In those cases, you may need to determine success using something other than how well the forecast predicted the actual values.

Data!

Once you have a well-defined problem, you must identify what data you need. In some cases, the data you want may not exist. What can you do? You might be able to request a change to processes so that your organization starts collecting the data you need. In other cases, it may not exist within your organization, but it may be something you can get from somewhere else (for free or by purchasing it).

Once you find the data you need, you've got to ensure you have contractual rights to use the data and the technical permissions to access the data wherever it's stored. After you've located the data and have permission to use it, you also need to consider *how* you get the data. Often data scientists are not analyzing the data in the same place as where the data "lives," so you've got to figure out how to get the data into the tools you want to use to analyze the data and build models. (Organizations may

not want analysis being done on production databases for a variety of reasons, such as slowing down clients who are using the same database or the potential to accidentally overwrite or delete the original data.) The specifics of how you access your data often depend on where your data is hosted. Sometimes data collection requires accessing the data via an API, other times you may have to import the data from a CSV, which could be the output from a SQL query.

Identify toolsets

There are many tools you can use for data science. Some you must license and pay for, while others are open source and free. I'm a fan of using Python for data science, especially Jupyter notebooks for the early stages of data and model exploration. It's a great option for being able to document your approach and share the results with others. If you don't know how to program, there are options like Azure ML Studio, which is a great click-and-drag option for playing with machine learning algorithms. There are a ton of great resources online for learning to use these tools and becoming your own citizen data scientist.

Present your results

Once you define your problem, collect data and create a model, how do the results get incorporated into your workflow? The entire purpose of building the algorithm is to get to the point where you can communicate the results to facilitate data-driven decisions. However, this can be one of the larger barriers that data scientists face.

Typically, results are returned in one of two ways:

- communicated to the stakeholders via reports and dashboards, or
- communicated to services.

For the first phase, most likely you'll just be creating reports or dashboards to share. (Remember, you're starting small.)

At this point, most results are displayed in presentations. The difficulty with a slide presentation is that it is static reporting, and when questions come up about the analysis ("What if we filter down just to this time range?" "How does that affect the results?"), you likely have to get back to the stakeholder later with updated results, unless you predicted those questions would come up and prepared variations in your analysis.

While this is OK in the early phases of the data science workflow, this isn't where we want to end.

So, let's move onto the next phase.

We're Walking!

Where do we go from here?

You now have your toolsets identified for development. You developed your first model. You built some confidence that machine learning can solve some of your organization's problems, and you're starting to get more requirements for new model development. Now what? It's time to Walk.

There are a couple of issues that start to crop up at this point.

You've got to figure out how to build out a team to start handling all the new requests. We'll talk briefly about some of the options you have for building a team, but that could be a whole topic of discussion itself.

Once you build a team, you must figure out the larger workflow process. You'll have multiple people working on the code base, so you'll need to figure out code management strategies. You'll also likely need to figure out a review and approval workflow, as well as some quality gates to ensure the models you're deploying meet expectations.

As you start producing more models, you also need to start thinking about how you're going to get the results of all these models into your stakeholders' hands. It would be a shame for all the hard work your team does to get stuck in the "data science lab" and never get used. Unfortunately, that is another barrier that many data science teams face. Ideally, the results will be available in an interactive format that responds to changes, but that may not always be possible. Let's aim for that goal though.

Building a data science team

How you build out your data science team depends a lot on the structure of your company and how much open communication there is between teams. Some suggested configurations are listed below.

centralized	matrixed	embedded
Dedicated team of data scientists compartmentalized from the rest of the company.	Dedicated team of data scientists where individuals are temporarily assigned to other teams, but return to work on special projects.	Data scientists are permanently part of cross-functional teams.

Keeping your data science team centralized allows for greater cohesion of the data science team and an increased sharing of best practices. It's easier to ensure consistency with coding styles, code management, and quality requirements, as well.

Embedding data scientists allows for greater cohesion with the team focused on the specific project, though, and can help provide the contextual knowledge necessary for successful feature engineering and the building of quality models.

I like having a team that is somewhere in between centralized and matrixed where the data scientists are partially assigned to another team during the project lifetime but do not completely "leave" the centralized data science team. This helps ensure continued collaboration and discussion with fellow data scientists.

Model and code management

Inevitably, once you have more people working on code, you need to have some sort of model and code management system. If you're not familiar with code repositories like Bitbucket or GitHub, they allow you to store your code in a central location and help guide your team through the development workflow.

These repositories feature version control to allow your team to track the changes made to the code base over time (and easily revert to previous versions if necessary). They also use access control to restrict who has access to the code and workflow tools to guide the code review processes through pull requests.

Make quality a priority

As your team picks up momentum and expands the development of models–especially if your organization comes to rely heavily on a "citizen data science" model where those creating the models may not have a data science background–it's incredibly important to have a well-defined code review and quality assurance process.

Having a codebase that is maintainable and understandable is incredibly important. One way this can be achieved is to ensure the code is fully documented using a well-defined documentation style. A reliable codebase can be obtained through the implementation of peer reviews that look for both technical and logical correctness, as well as by developing unit tests that must pass successfully before the code is submitted for review.

In addition to having a well-documented, reliable codebase, you also want to make sure your models perform well. In the Walk phase, we talked about how to define whether a model is successful or not. As the team grows, there needs to be a well-defined evaluation process. Standardization of the evaluation process ensures all models are assessed consistently.

Operationalize models

There are many ways to operationalize your models. Which one works best depends a lot on how your organization is set up (self-hosted vs cloud-hosted) and exactly what you're trying to do with the results of the model (sending the results to another service vs end-user directly accessing the results).

For algorithms that have pre-determined slicing/dicing possibilities, you may want to integrate the models directly into the ETL (extract, transform, load) process and make the results available alongside the source data to use in reports and dashboards. For models where it would be beneficial to have the ability to change input variables on the fly, you may want to use microservices to be able to make requests in real-time and see different results in your report/dashboard platform.

We're Running!

We've got models!

Once your organization is producing a lot of models regularly and has models that are in production, there needs to be a strategy for how to maintain them, stay aware of their performance, and make changes as necessary. When this happens, it's time to Run.

Of course, this isn't the entire list of things you need to consider, but some of the things we'll focus on in this phase are:

- continued quality assessments,
- governance for central awareness and accountability,
- care and maintenance of models, and
- user feedback.

While many of these topics come into play during the Run phase, it's important to be aware of these things during the Walk and even Crawl phases. Knowing that you'll need to address them in the future will help you determine how you want to architect your data science workflow early on.

Continued quality

Part of the model assessment happens before you release the models. You've got data to test the models on and hopefully before you release, you're getting good results. But you're not done after you deploy the models. Model performance should continue to be evaluated as new data is added and as the number of models in your portfolio increases. Implementing an automated alert system to watch for evaluation metric deterioration is an excellent way to monitor performance. Poorly performing models should be assessed and updated or removed if they have reached the end of their lifespan.

Planning for updates is important. For some models, the data may not change very often, and updates may not need to be as frequent. Some updating can be planned, but other models will need to be updated for unplanned events that create large-scale change. Events like the recent COVID-19 pandemic most likely drastically affected many models that depend on human behavior.

The pandemic may have far-reaching, long-term effects on data models. Determining whether the pre-pandemic data will continue to be useful for training models, what to do with data collected during the pandemic, and how both will affect forecasts on future data is probably highly dependent on the particular context of the model in question.

Centralized governance

With all the models being used in production, there will need to be some centralized governance to be a single source of truth and guidance.

Governance should include a definition of the roles and responsibilities for each piece in the data science workflow, from model creation and validation to model approval, management, and deployment.

The governance should also ensure models and data usage follow legal and regulatory compliance. While this is not the most exciting part of the data science process, it is unquestionably an important part and can be a complete project showstopper. As part of legal compliance, information about who has access to the data and models (access control) should be tracked within the centralized governance. In addition to legal and regulatory compliance, a review for ethical compliance (also known as "responsible ML") should be a part of the governance to ensure bias is not being included in the models at any point from the problem statement, raw data selection, and feature engineering to the final trained model.

Documentation is incredibly important and needs to be available and complete for all models. Any assumptions about the model, information about the data sources, expected inputs and outputs, and how the models are evaluated should be documented. Having a centralized location for this information helps people know where to go and also helps to enforce a consistent documentation style, which helps ensure the documentation is easy to understand and users know what to expect.

Additionally, there should be a framework around documenting/tracking/communicating when changes are made to a model and by whom. (This may be a person or even an automated system that re-trains models on a set schedule.) The ability to trace a result back to a specific model (where the result may not have come from a model that is currently in production) may be necessary as part of legal compliance.

It is beneficial to have this information collected in a centralized place where there is knowledge of whom is using these models and might be impacted by any changes so those impacts can be communicated. Centralized governance helps when situations arise like COVID-19; the models can be evaluated to determine which might be affected and need to be updated. It also allows the organization to quickly figure out who uses the models, who owns the models, and who will be responsible for making necessary updates.

Care and maintenance of models in production

As discussed in the governance section, in addition to general model updates, there also needs to be some discussion around who is responsible for managing the deployed models. Many of your models will change over time, and as an organization, you'll need to determine a strategy for how many versions of the model you will continue to support and what the process will look like when you deprecate models. Deployments often need to be coordinated with other teams to ensure that model versions match what is expected by the services/dashboards/reports utilizing the models.

Once the models are in production, someone (or something) needs to ensure they are up and running. While you could wait for an end-user to report that your service is down, it may be possible for you to set up an alert system to send your team a notification if there's a system failure. Depending on how large your organization is, tracking your services may be the responsibility of the data science team or it may be a dev-ops team that tracks the services in a larger organization. In addition to the general status of the model, there should be a determination of who is responsible for ensuring the services run and the models scale appropriately.

User feedback

It's important to remember the end user's feedback should be part of the data science workflow too. If your organization is developing great models and they aren't being used, it's important to figure out why.

One of the first steps is to identify which models are being used and which aren't being used. Check with the users to see why models aren't being used. In some cases, it could be that you have "bad models." What defines a bad model? It could be that the models aren't performing well at their designed task (hopefully this is something that could be detected before we get negative user feedback if you've got a good evaluation system with alerts set up). It could also be that they don't answer the question that they were intended to answer. At this point, we may have to start the process over. And remember that's OK! Building models is an iterative process.

Maybe your model isn't bad though; it might be performing wonderfully when you look at evaluation metrics. So, why isn't it being used? Sometimes this happens because users don't know these models exist. Sometimes this happens because despite how good the model is, the end-user may not understand what the results mean or exactly how to use the results.

Documentation with examples and human-readable explanations are helpful in this case. Brown bag learning sessions or other training methods to help people use your models can improve utilization and satisfaction with the results.

Training is also a great way to make sure models are being used in *expected* ways. When models are applied incorrectly, it can result in users being frustrated with the results and might cause them to believe the model is performing poorly when it isn't the model itself that is the issue.

The Crawl, Walk, Run paradigm is only one of many ways you can introduce data science into your organization. The great news is there's a lot of flexibility in how you implement each of these steps, and you can break the process into more phases or combine them into one phase if you want. Whether you're already using data science in your organization, starting to build a team, or learning to build models, hopefully, you were able to gain a few nuggets to enhance what you're already doing within your organization.

Ashley Pitlyk, Ph.D. is Director of Data Science at MedeAnalytics.