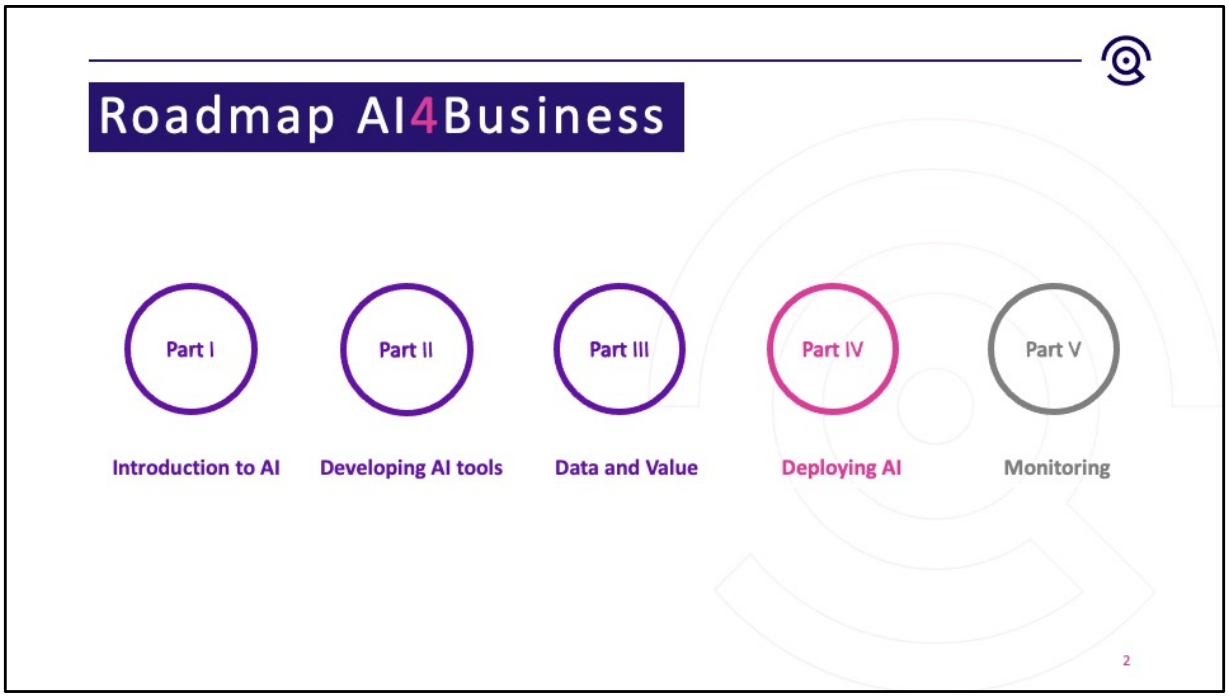
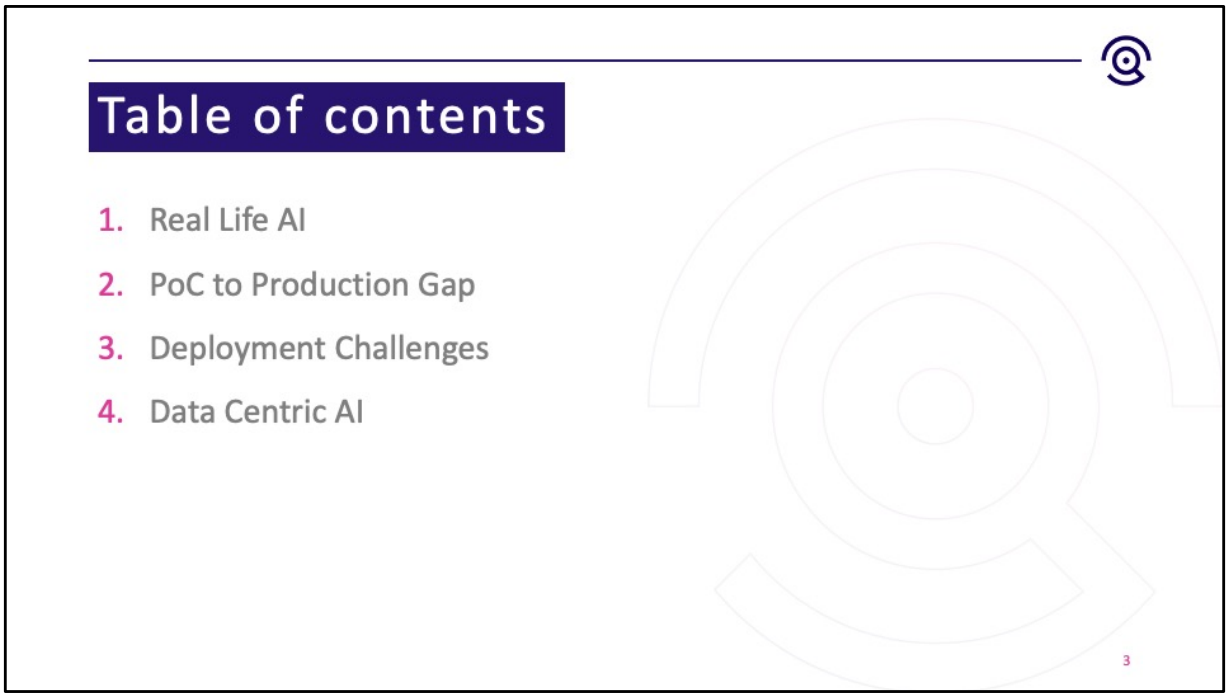
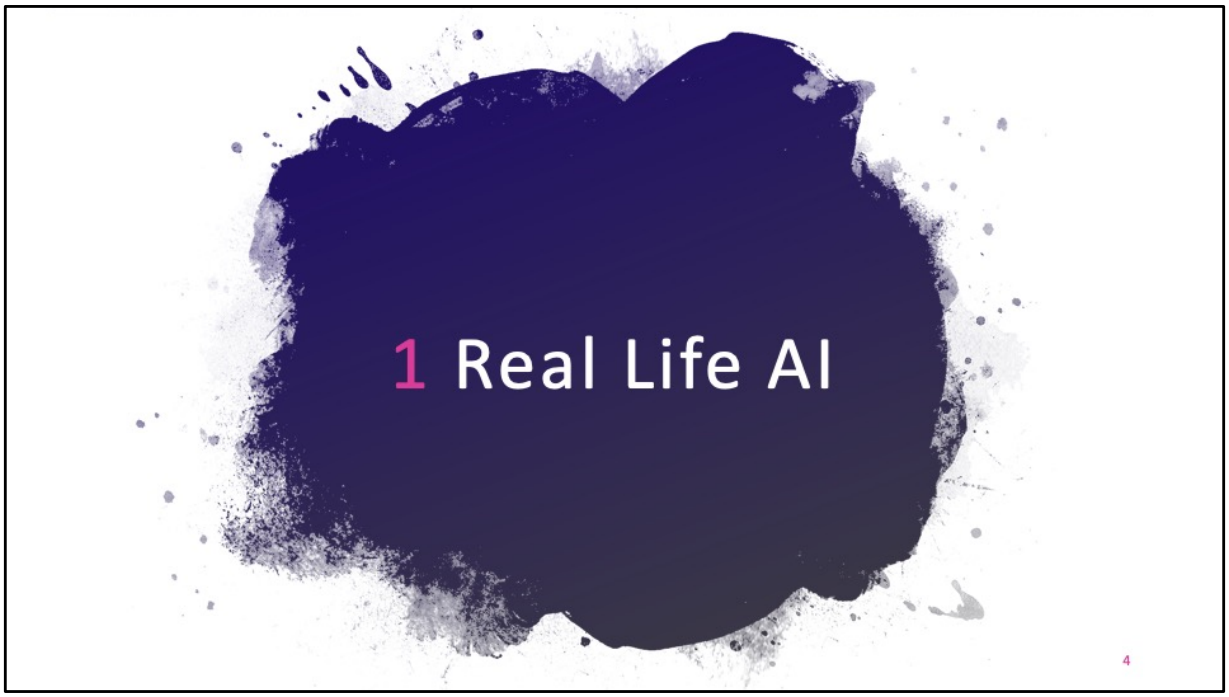
Welcome to the fourth module of the AI4Business course on deploying AI solutions. Today we will lift AI from the data scientist's playground by putting AI into production environments where it can actually start producing real business value.



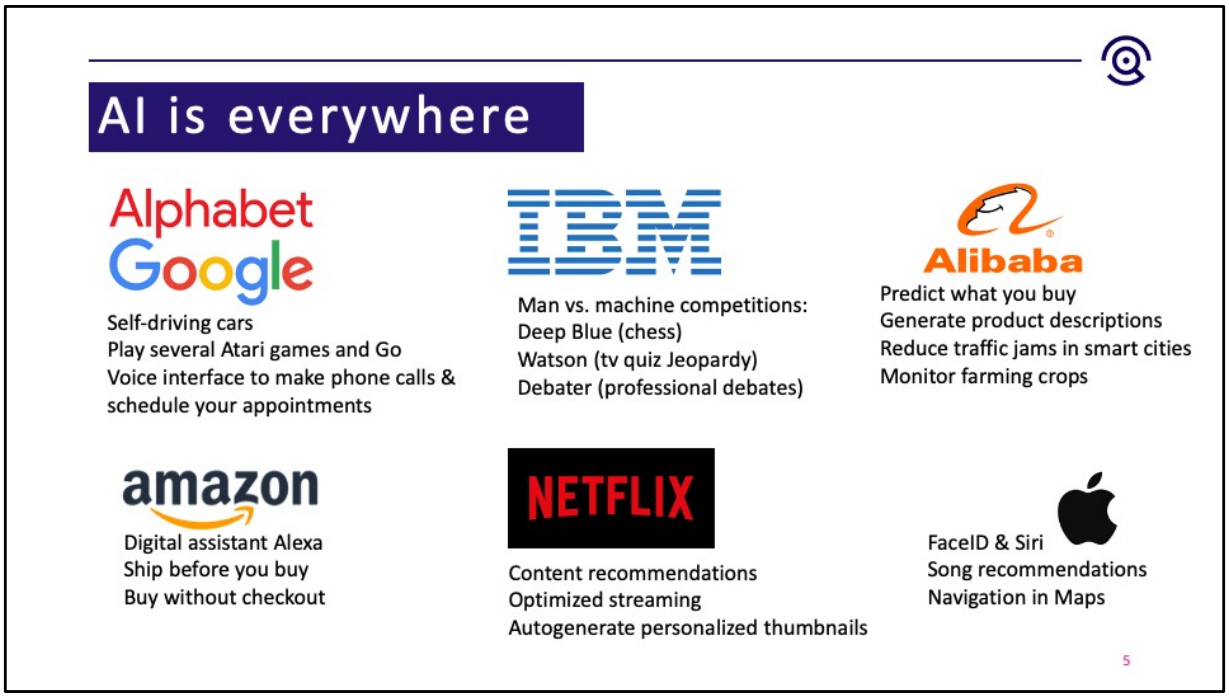
This module is the fourth out of five modules in the AI4Business course. We have seen how to develop AI tools and how to capture value with AI. Although not strictly necessary, reviewing the previous 3 chapters will ensure that you are able to make the best out of the content we are going to share about how to put those valuable AI solutions into production.



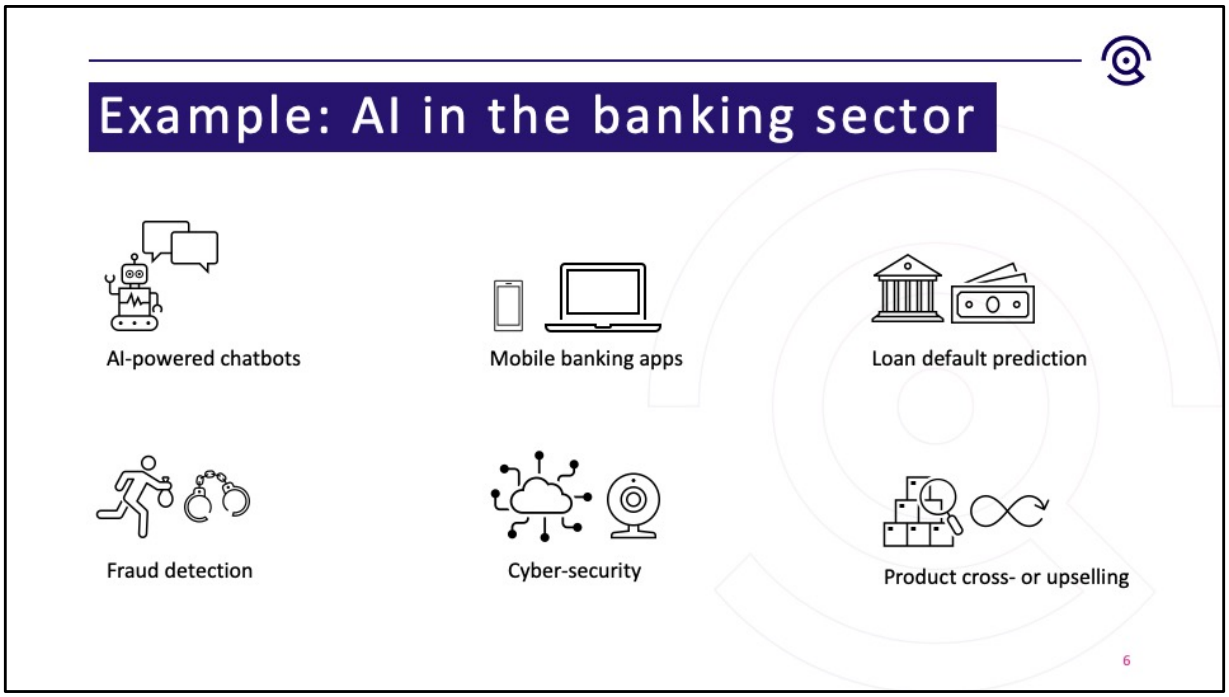
Today we cover the challenges of bringing AI to life in the real world. We start by providing some examples of AI in real life and why it is so different to other software solutions. Next, we shed light on why it is so difficult to go from a proof of concept to a fully functionable AI application. Afterwards, we highlight the most common deployment challenges that AI teams face when trying to deploy business algorithms. We finish this module with an overview of the new Data centric AI paradigm.



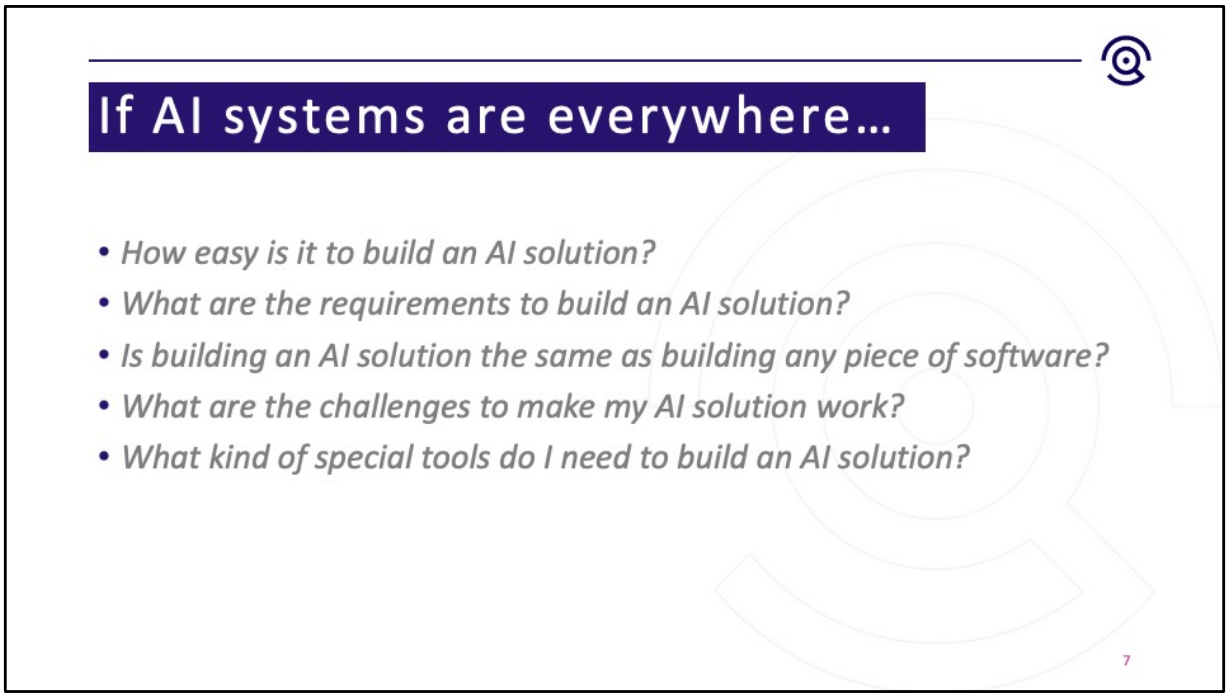
Let’s start with a discussion of AI in real life and the kind of difficulties that one might run into.



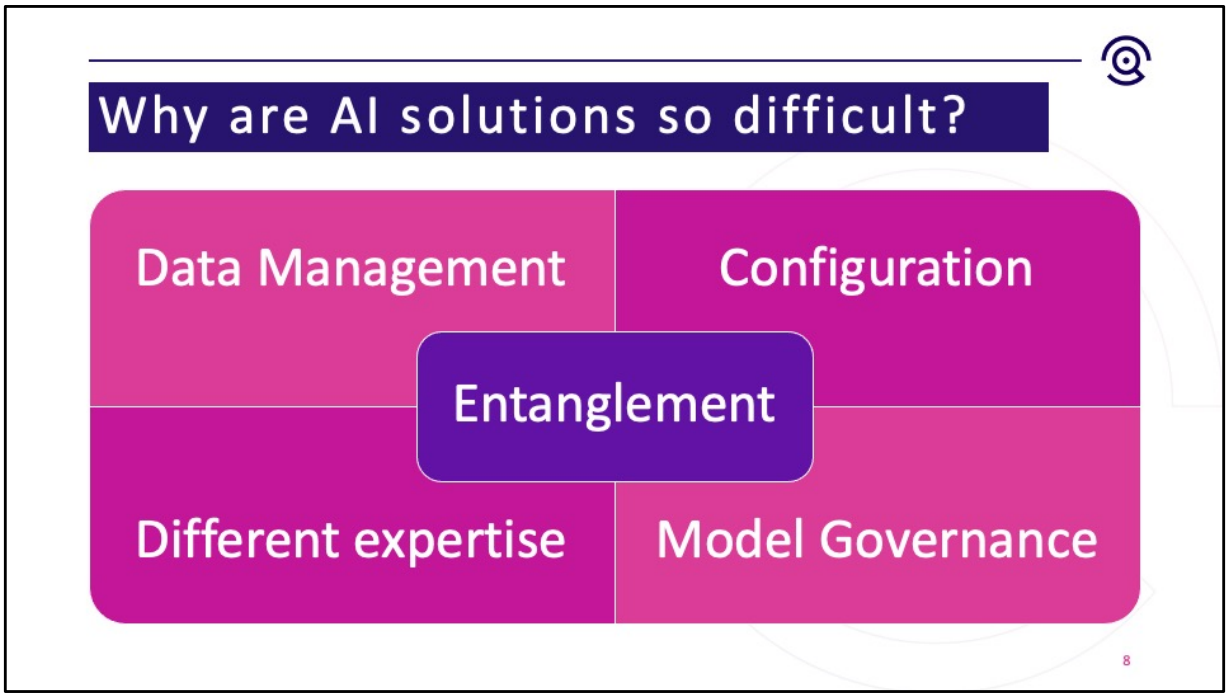
AI is everywhere and everybody is using it, maybe even without realizing it. We list some examples of big tech companies and how they apply AI in their products or daily operations.



This is just a small example to show how omnipresent AI is in the banking sector for example. AI is used for chatbots, mobile banking apps, loan default predictions, fraud detection, cyber-security and product cross- or upselling.



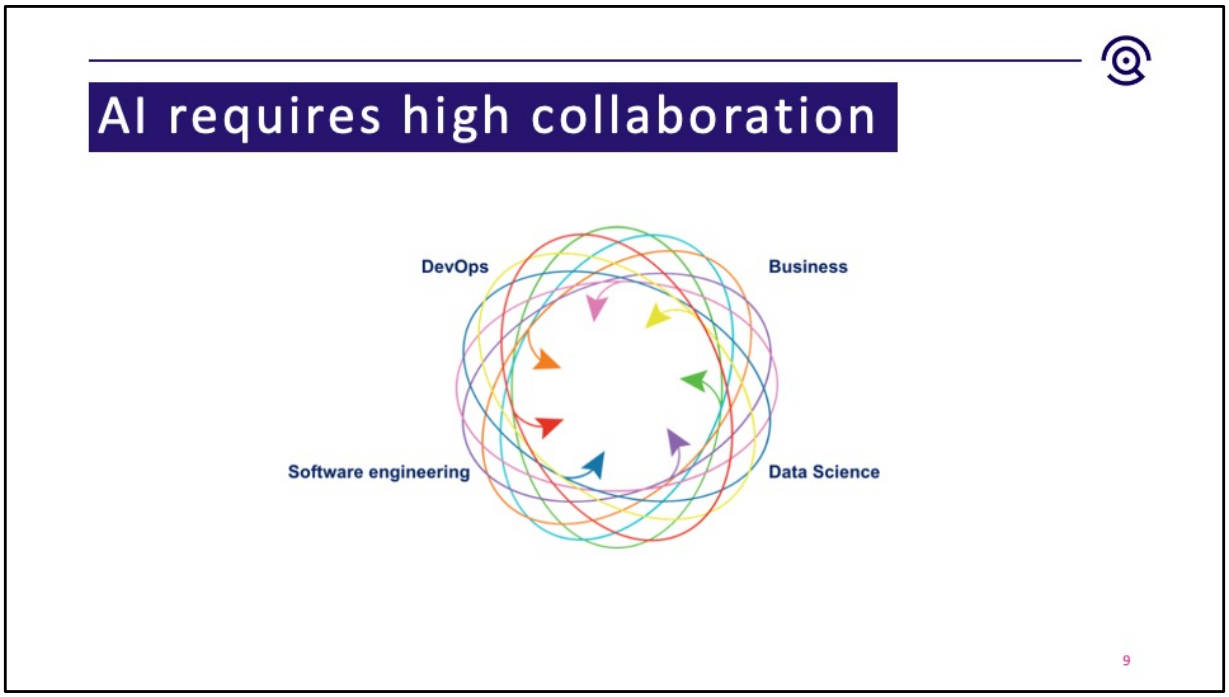
AI systems seem to be omnipresent. So it is quite natural to start asking questions about what it takes to build those solutions. Some possible questions are the following:
-How easy is it to build an AI solution?
-What are the requirements to build an AI solution?
-Is building an AI solution the same as building any piece of software?
-What are the challenges to make my AI solution work?
-What kind of special tools do I need to build an AI solution?
We aim to answer these questions in this module!



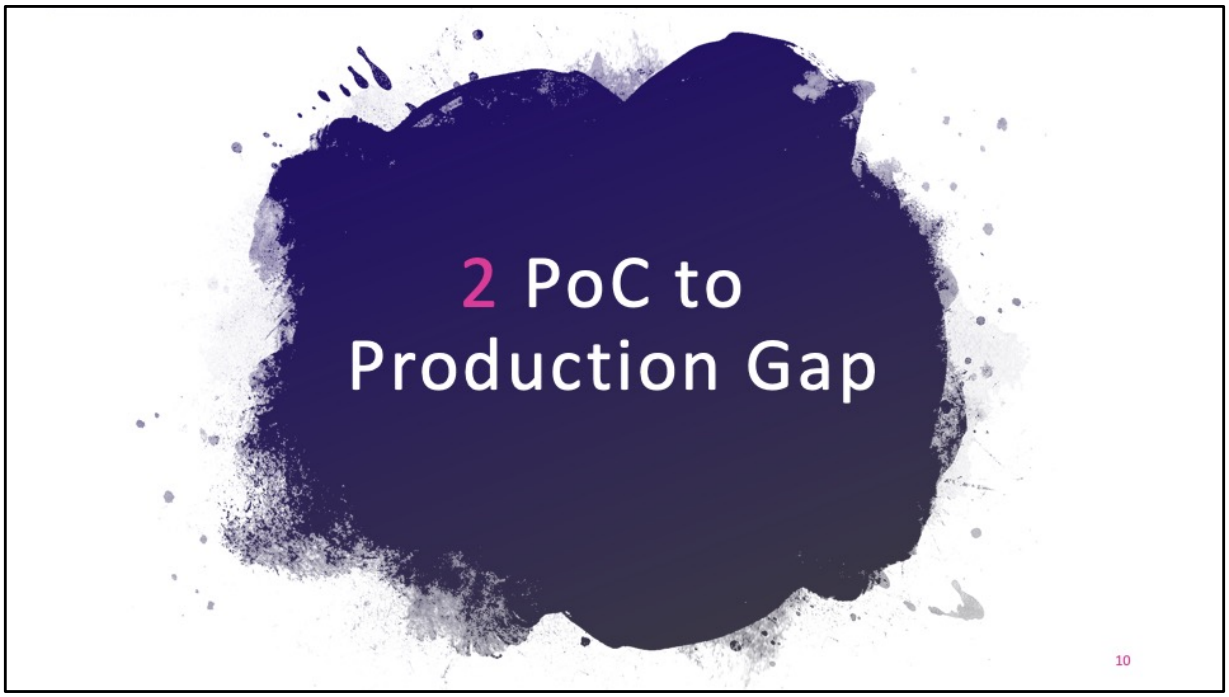
The simple answer to the previous questions is that AI systems are rather difficult to build properly as they require particular attention to some topics and points of the project lifecycle. From a high-level point of view, most difficulties in building AI systems arise from the following 5 places:

- Entanglement: meaning that “changing anything changes everything” . There are simply so many things to track and they are so interconnected that if changes are not properly tracked, even the smallest one could spell significant trouble down the line.
- Data Management: good data governance is key to building any AI solution. As we do not only need to maintain code but also data. Tracking all data dependencies, creating robust data pipelines, having good metadata are only some of the few things that a good data strategy requires.
- Configurations: when we think about configurations in this context, we think of how ML models are constantly being iterated and might contain a vast number of parameters that should be tracked. However, data transformations and other specialized components might have parameters that interact with the AI system as well. It is for that reason that it is key to have flexible and well-maintained configurations.
- Different expertise: AI solutions require interdisciplinary teams to cooperate

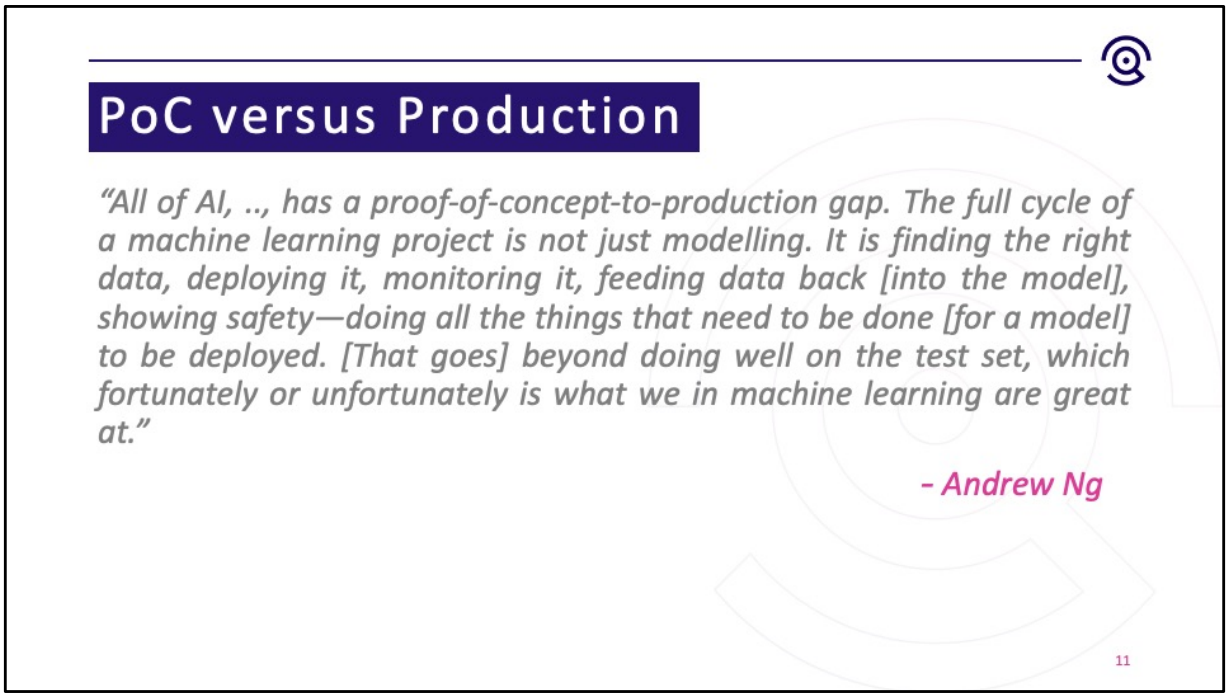
which can result in no single person or team understanding the totality of the solution. If not managed well, teams will start blaming each other for mistakes without necessarily being able to understand where failures are arising from.
- Model governance: On top of code and data, AI system also have models as artifacts. Those models should also be subjected to proper governance. Deploying the wrong model or training a model on the wrong data might lead to serious failures in production.



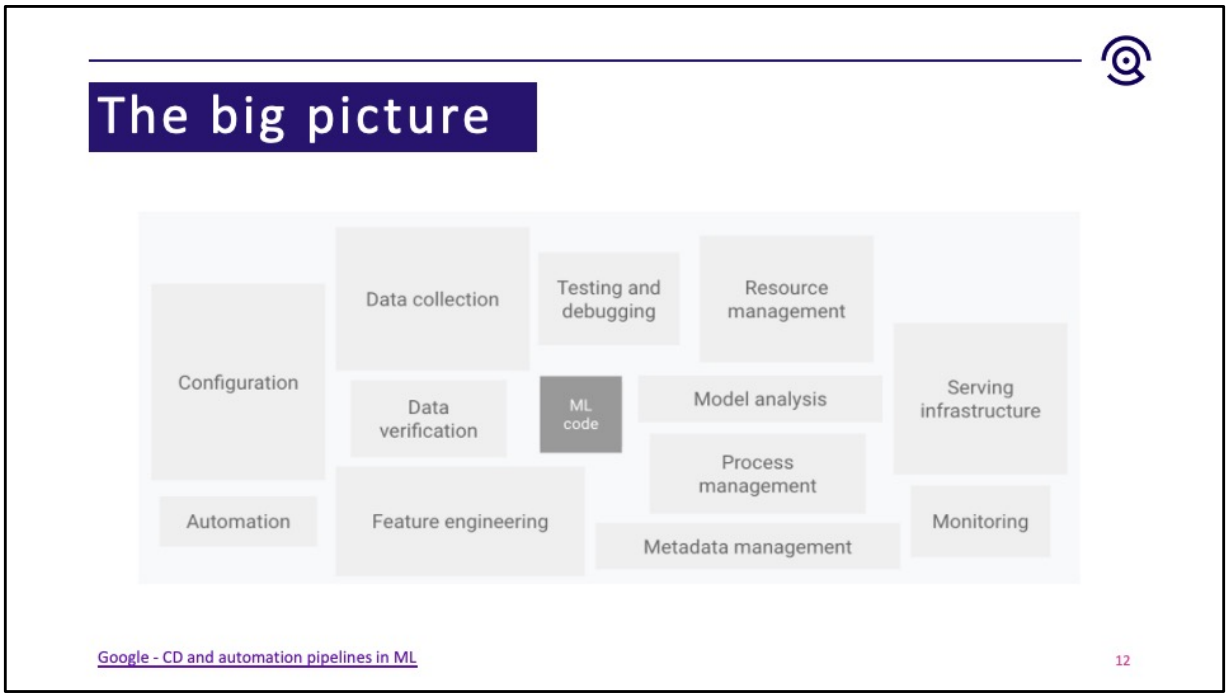
Finally, as mentioned before, building a good AI system requires a high diversity of expertise. Preferably cross functional teams working together towards the same goal. Real life AI systems are very different from POCs and purely academic or experimental set ups. When building a real life AI you not only need the data scientist to bring forward the Machine Learning expertise, but also a business counterparty to specify requirements and make sure that actual value is being created. On the other hand, you will also good need software engineering to build a robust product as well as DevOps (or more particular to this case, MLOps) to ensure the appropriate design and continuous functioning of the solution. During the rest of this module, we are going to expand on the challenges of building AI solutions from technical aspects to business problems.



Let’s see what it takes to go from a proof of concept (or PoC) to an actual production model.



Andrew Ng is a Stanford Professor and founder of deeplearning.ai and a big name in the AI world. This quote from him summarizes what we typically focus on in machine learning and what we tend to overlook.



When we think about AI or Machine Learning, we tend to focus only on the model. This is the part of the solution that allows us to generate predictions plus the code that helps us generate their outcomes. This is also the part that is more easily recognizable, not only by the data scientist but also by the business. That is because the model is at the core of the functionality and one can easily explain it without having to think about all the operational hazards that are needed to actually make a model available in a production environment. In other words, the Machine Learning code is just a tiny fraction of the end-to-end solution.

In a PoC one would probably only need this part (the ML code) to be able to show some results. But when it comes to production, the picture becomes way larger. Many new elements start to pop up.

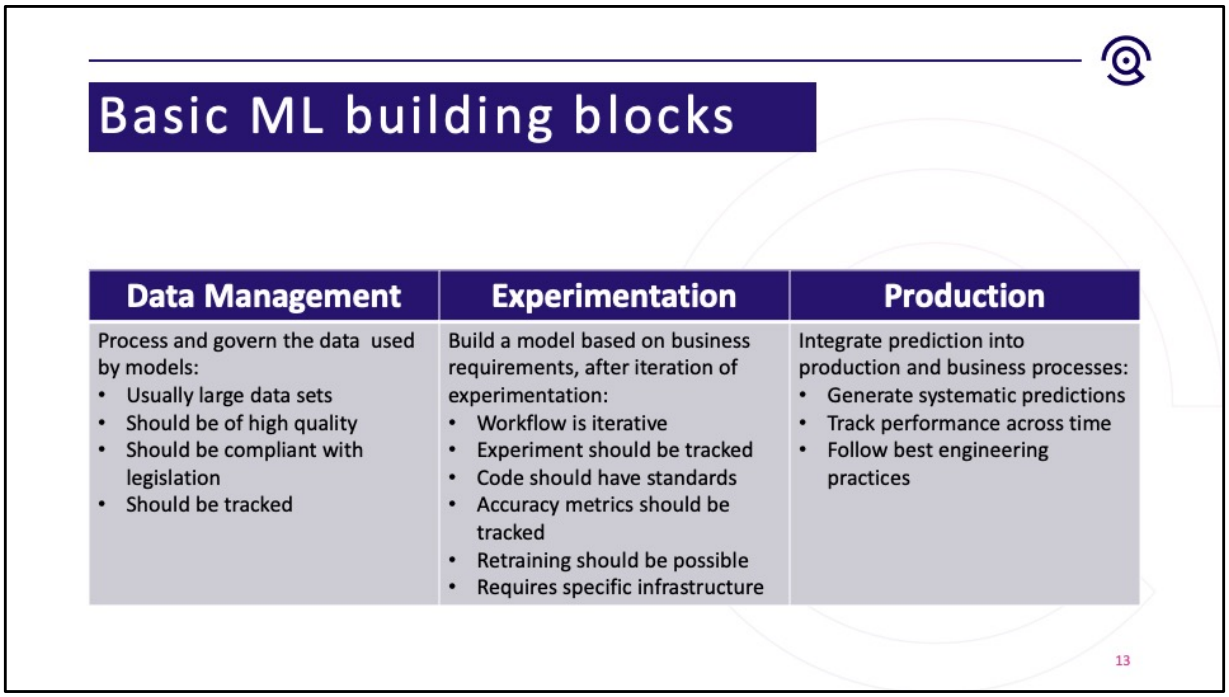
Data collection and verification become part of the process for example.

But it is not only about the data. There is a lot of configuration overhead involved as well as the burdens of having a fully automated solution.

Furthermore, things like monitoring of both the model and resources become a part

of the picture.

In short, ML solutions are large and complex, and the code that implements the core of the ML functionality is just a small part of the puzzle. This raises many challenges towards deployment, especially because most AI experts or Data Scientist design their solutions as a PoC, often neglecting the complexities of real world production environments.



When we think about the basic workflow or lifecycle of any ML application, we could easily identify 3 large building blocks. Namely Data preparation (or more generally data management), experimentation (or model building) and production (or deployment of solutions).

Data management concerns everything related to the processing and governance of the data required by the models.

- In ML solutions these are usually large data sets.
- And in order to generate trustable results these data should be of relatively high quality, otherwise it is just garbage in garbage out.
- Data should also be compliant with current legislation (for example GDPR). Different data might have different levels of sensitivity and it is not always possible to use it as is or even at all. You need to keep this in mind when building ML systems.
- Data should also be tracked and versioned, since we need to be able to always reproduce our results. Not being able to reproduce past results highly erodes the confidence in your AI solutions.

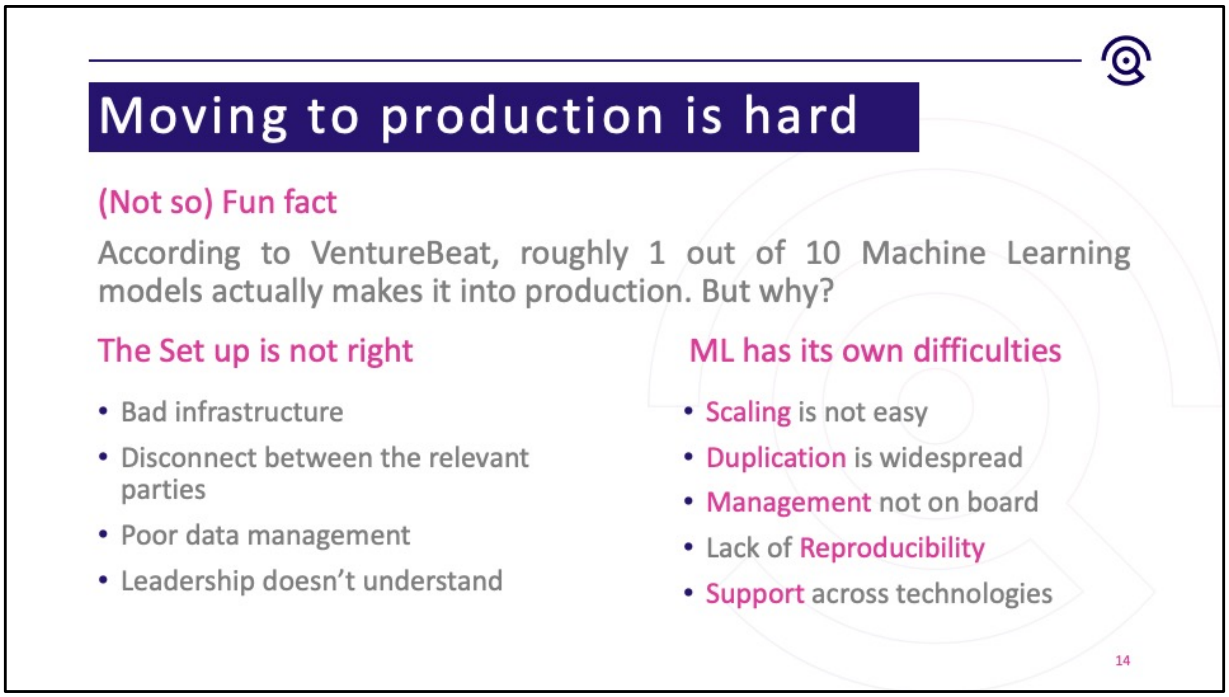
Experimentation is that second stage when it comes to building ML systems. We iteratively build models based on business requirements and specifications.

- The iterative component of the workflow is very important as it allows us to

systematically improve the quality of the solution.
- It is for this very reason that these experiments should be tracked, as we want to know the reasons of the improvements as well as being able to roll back to previous versions if needed.
- Also the retraining of models should be easy to achieve, which is only possible if we keep in mind that results must be reproducible.
- One thing to keep in mind is that building adequate ML solutions requires proper infrastructure, sometimes this means software, but other times this is about processing power or even just simply well-defined processes.

Production is the last part of the workflow in an ML project's lifecycle. Moving to production entails how do we bring everything together in order to:
- Generate systematic predictions either offline (batch) or online (real-time).
- Track the performance of those predictions across time and make sure it is not deteriorating. Also to act accordingly in case performance is degrading over time.
- For this we need to follow some engineering best practices (specially DevOps) which are not so straight forward in the case of ML systems.

This part will focus on this last point and shed some light on why it is so difficult to move ML solutions from PoC to production environments.



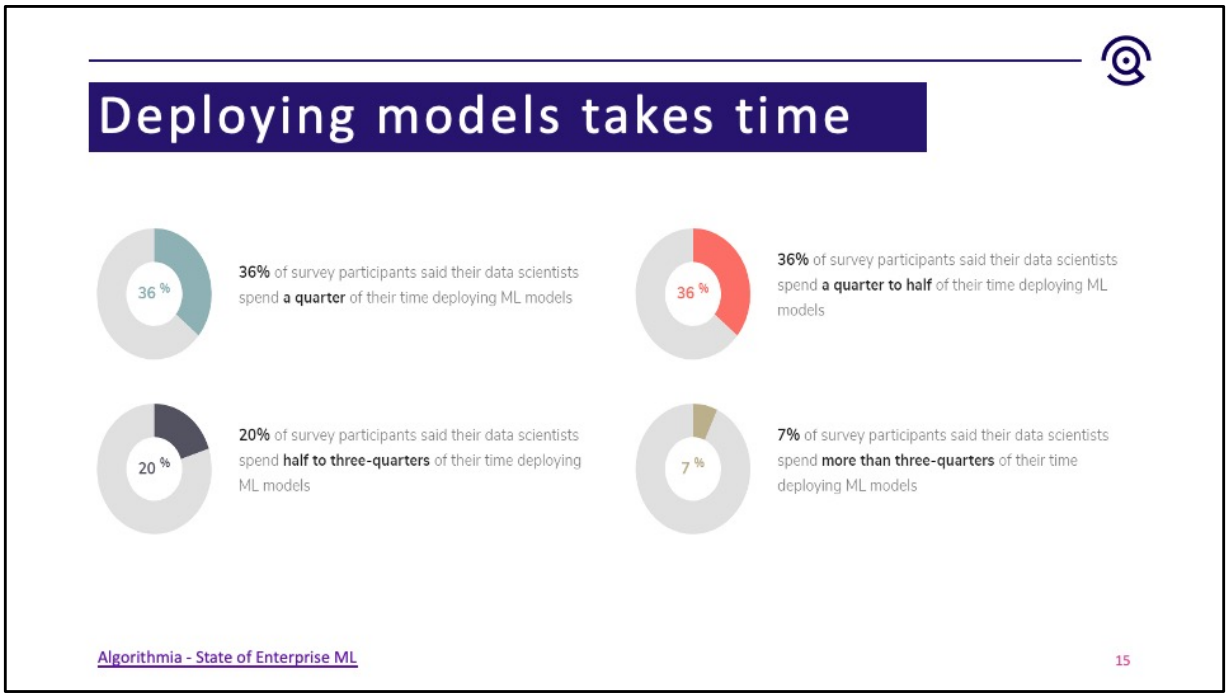
One of the main issues challenging the adoption of AI is how difficult it is to move from a proof of concept to a real production environment. You might be surprised that, according to VentureBeat, only roughly 1 out of 10 machine learning models ever makes it into production. Which means that only 10% of the efforts produced by Data Scientist ever translate into real business value. The main reason for this disproportionated rate of failure is that the set up of the organization is not right to perform advance analytics properly. Some examples of bad set-ups are the following:

- The infrastructure is not up to par with the requirements of the analytical problem that needs to be solved.
- There is a disconnect between the parties involved, meaning that all the business, IT, Data Science and other operational efforts are not well orchestrated.
- There is poor data management in place (for example when it is difficult to access relevant data or data governance is not there).
- But in many cases the main problem is that the business does not understand what exactly is required for a data science team to succeed, as leadership is more than just allocating budget.

However it is not only about the set up, Machine learning comes its own set of challenges:

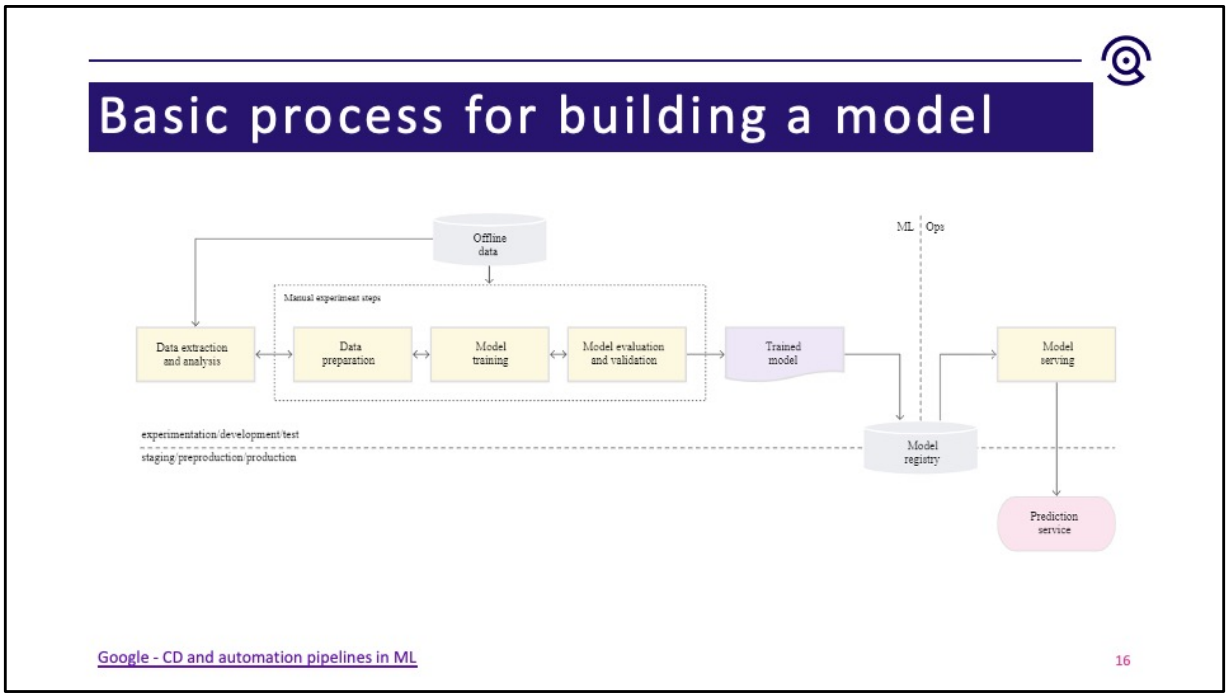
- Scaling Machine learning systems is not straight forward.

- There is a lot of duplication coming from the disconnect between data science and operational teams, which in term translates into wasting valuable resources.
- Also, management is not always on board with analytical solutions. There is always resistance to change, because things have been done the same way for a long time.
- Another issue is the lack of reproducibility, as many data scientist are often not familiarized with good engineering principles.
- And finally, good support across all of the technologies within an organizations is not always a given, with Machine Learning pipelines often interacting with a wide variety of frameworks and software.



It is not only that many of the models that are developed never makes it to production, but on top of that, the ones that do get there can take a significant amount of time getting there. This is particularly true for those organizations that do not have the adequate infrastructure or even operational process in place. According to a survey by Algorithmia, over 63% of data scientists spend more than a quarter of their time deploying ML models. And as much as 27% spend more than half their time on these deployment efforts.

This is particularly worrying, especially when we know that most of the models do not make it to production anyways. With the exception of very particular cases, the data science job description does not references this part of the job either, which in many cases leads to high turnovers on data science teams. Data scientists encounter themselves in this situation where the job they end up with is very different from the one they thought they were applying for, leading to a drop in motivation.



As we have discussed before, building a model has certain steps that need to be flowed through. Most data scientists would at least have a very simple workflow in which a significant portion of the workflow is manual labor.

From the picture we can see that probably every step up to the creation of the final model would be a rather manual process, and would go as follows:

1- First comes the data. The data scientist will connect to some data repository or database and extract the required data.

2- Next the data scientist will analyze the data to make sure it is of good quality and to understand its properties.

3- Then follows the data preparation so models can use it. This is the beginning of the iterative experimentation process, as different models might need different transformations of the data or feature engineering processes.

4- Once the data is in shape a model will be trained.

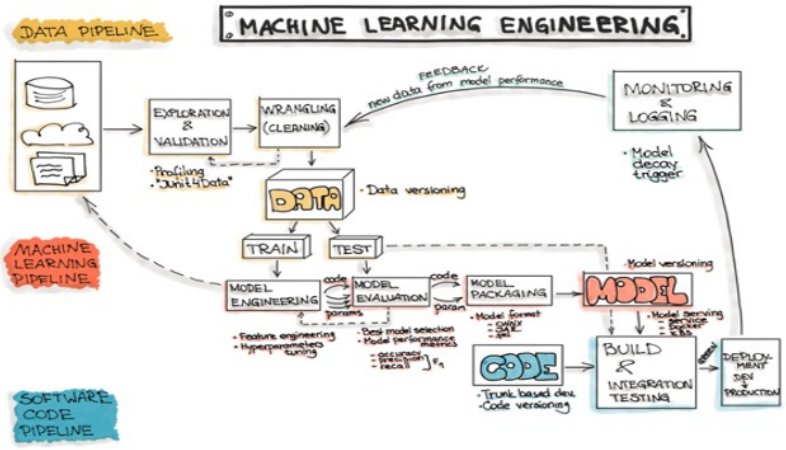
5- After which the model shall be evaluated and validated.

At this point we would either have found a model one is comfortable with and end the experimenting cycle, or go back to either retrain a new model or prepare some data again to train a new model.

In most basic operational set ups, this final model is then deployed into production. However, many problems will soon start arising, like lack of reproducibility or

auditability, no transparency, impossible to monitor the performance, high downtime for applications when something is not alright with the model. Not including all other difficulties that come with versioning data and the propagation of changes. These problems arise because real production environments tend to be rather complex or just not optimally managed.

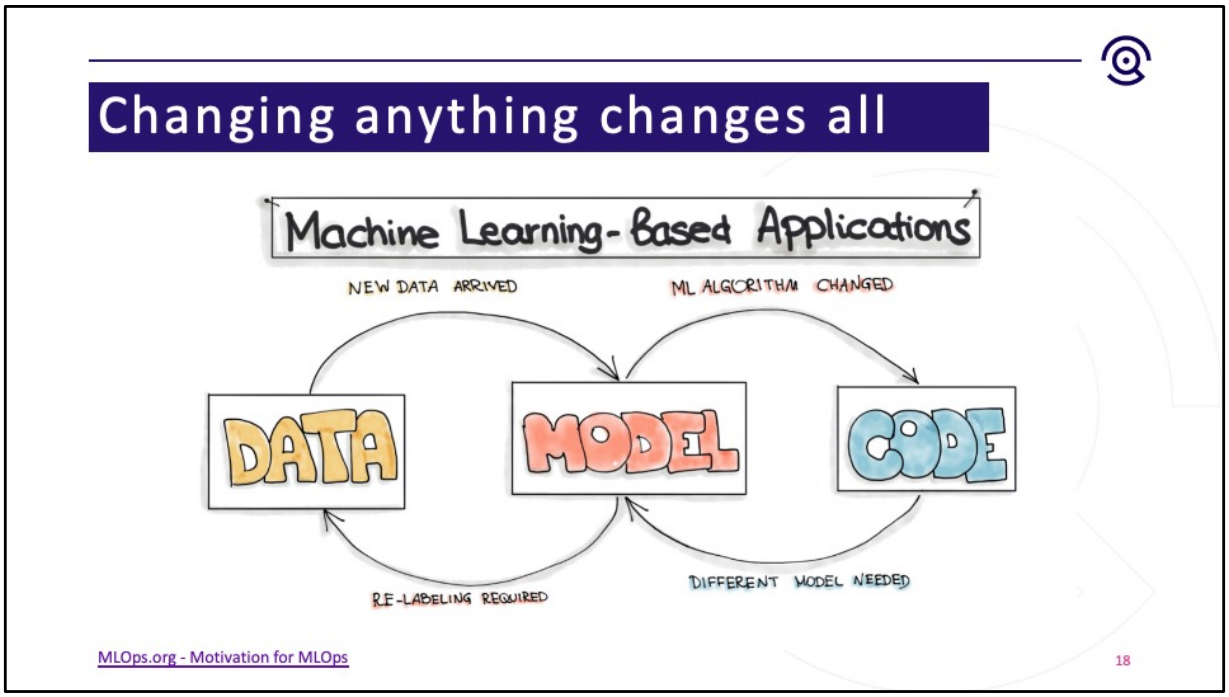
Real life is a bit more complicated



MLOps.org - End-to-end workflow

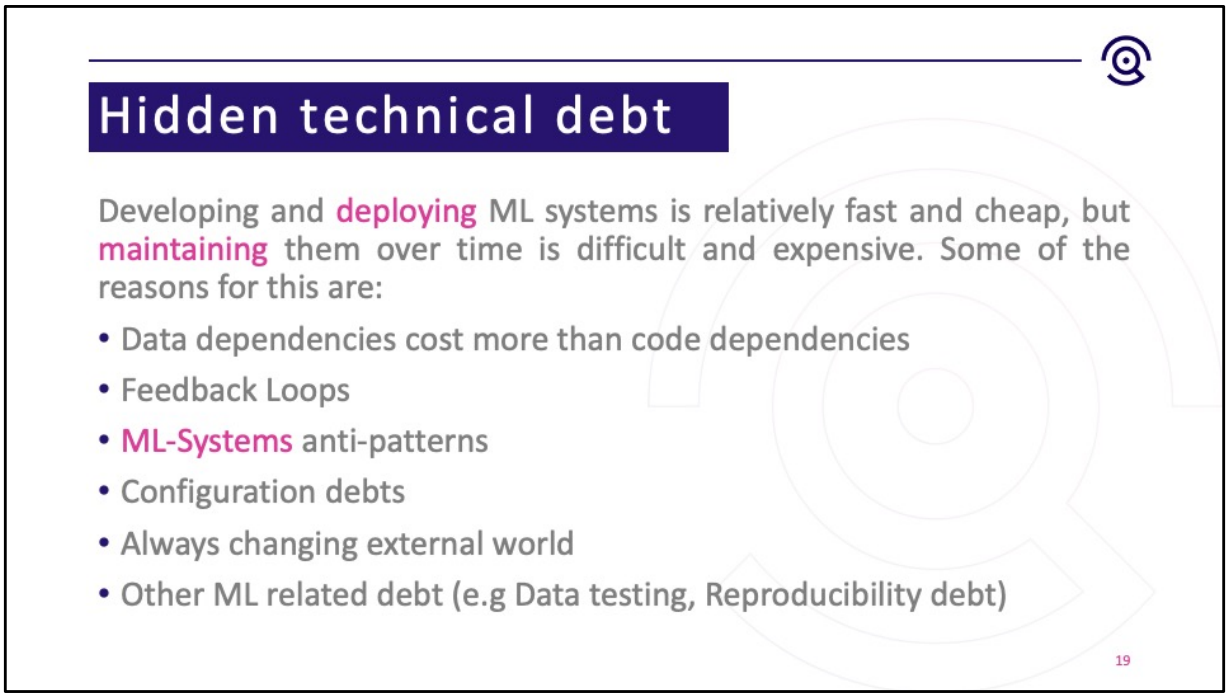
17

Here we can find a bit more realistic view on how and end-to-end workflow of an ML system looks like. There are three pipelines to consider, the data, ML model and software code pipeline. Although this picture is still a simplification of reality, we can observe how much more complex the situation really is. Especially because of the experimental and cyclical nature of ML systems. There are many things that need to be versioned, processes to be established, different kind of performances and feedback across components to be monitored. Also, unlike in typical software engineering, we not only need to worry about code but also about data and all sort of artifacts related to the models. Moreover, the cost of making changes is higher, as it is not only about compiling code or running a new version, but one also needs to re-do the whole experimenting cycle if a change in the model should be made. Later in this course, in the fifth and final module, we are going to talk about MLOps, which is a set of principles designed to help us deal with the added complexity. There we will explore deeper into the ML project life cycle.



So unlike in typical software development, the complete development pipeline has 3 components: the Data, ML Model and the software Code itself. This means that in ML systems, the need for an update might come from any combination between a data change, code change and ML model change. This is known as: the changing anything changes everything principle. This of course increases the maintenance complexity of such solutions. As even after already being deployed, we might need to still be able to make fundamental changes to our machine learning applications at any point in time.

For example, after a model has been functioning for a while, we realize the performance has decayed over time. We therefore might need new data to train a new model and replace the old one in the original solution. Another example could be the effect that an event like COVID might have over our models. In some cases retraining the models would be sufficient, but in other cases it might be necessary to re-formulate the problem to adjust to the new reality. Alternatively, it could be the business priorities that shift, so we need to take a different approach to solve the problem.

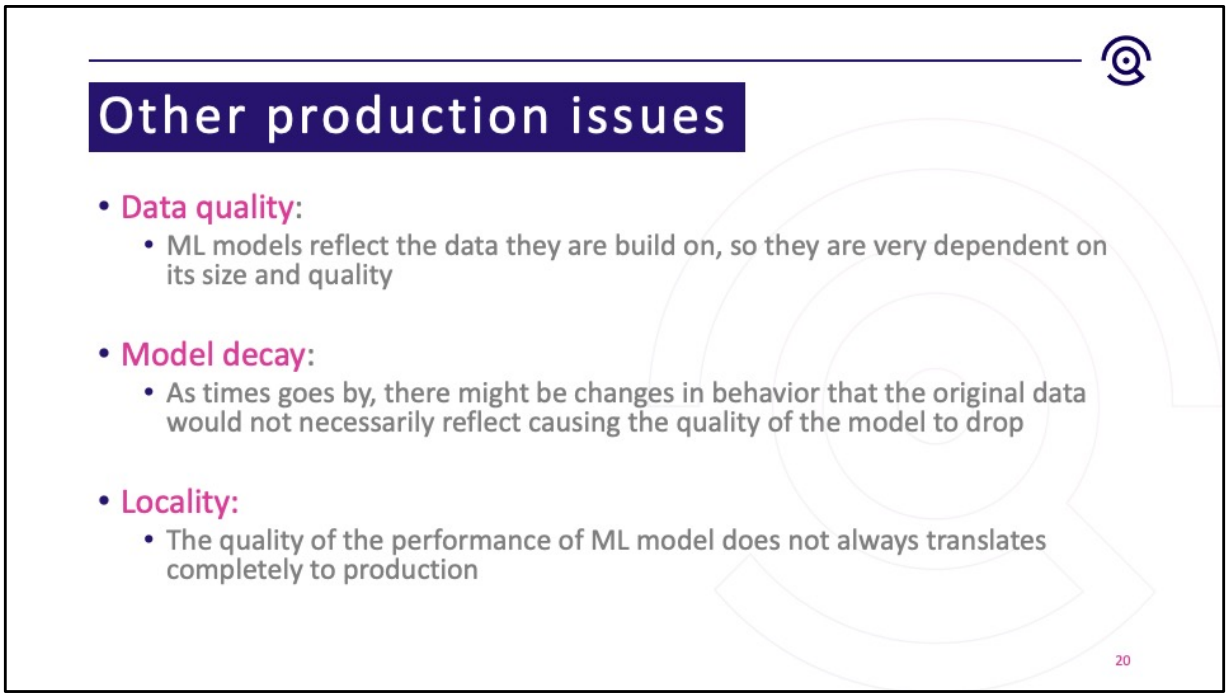


Another big issue with the move to production, is the huge technical debt that is hidden in ML systems. This makes not only their move to production difficult, but also their maintenance once they are deployed. In the last years, the democratization of software, education and ML tools have made developing and deploying ML systems a relatively fast and cheap ordeal in comparison with not so long ago. However, the maintenance over time is still very difficult and expensive. We can explore some root causes for some technical debts:

- Data dependencies cost more than code dependencies, and this is amplified by the enormous and ever-increasing amount of data ML systems need to operate on.
- There are many feedback loops in the ML workflow, as the systems often end up influencing their own behavior over time, and it is not always straight forward to identify where change is coming from.
- Surprisingly, only a fraction of the code in a ML system is really devoted to learning and predicting, as these systems are usually part of a bigger general-purpose solution. The problem is that, since most experts are not software engineers, a lot of bad software design patters (or anti patterns) start to appear, increasing the technical debt many fold.
- It is common for Machine learning system to have a lot of tunable parameters. Many Data Scientist and even Engineers tend to think of configuration as an

afterthought. For many complex systems that poses a huge operational risk, as incorrect configuration can greatly affect the outcome and be cumbersome to correct.

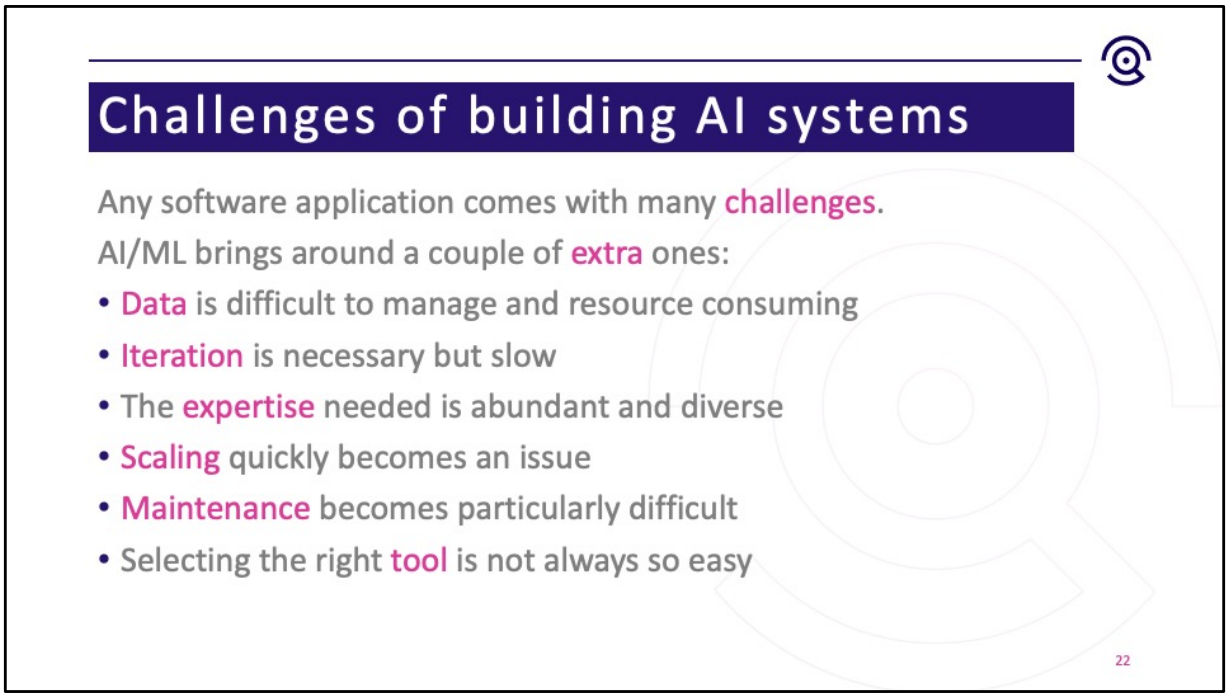
- One of the things that makes ML systems so interesting is that they are always interacting with the external world, but the external world is ever changing. For example, we can have fixed decision thresholds for decision making that have been manually set, but if the model is updated with new data or an unexpected event like COVID happens, then those thresholds might lose their validity.
- There are other types of debts that are directly related to ML, for example the debt generated by not testing the data to guarantee its quality; or the reproducibility debt. There might be even some cultural debt, as sometimes there is a hard line between ML research and Engineering, which does not allow to create healthy team cultures that prioritize the optimization of the processes and best practices related to the machine learning systems.



There are still some other issues going around in production environments:
- Data quality: ML models reflect the data they are build on, so they are very dependent on its size and quality. As mentioned before, if the data is not good enough, we would be in the garbage in garbage out situation.
- Model decay: As times goes by, there might be changes in behavior that the original data would not necessarily reflect, thereby causing the quality of the model to drop. Decay in model quality over time is a common problem, so monitoring your AI solutions is of the greatest importance.
- Locality: The quality of the performance of ML model does not always translates completely to production. For example, when transferring ML models to new customers, models which have been trained on different demographics might not perform at the desired levels or fulfil the expected KPIs

So long story short, when you have a working PoC ready that is showing promise to capture business value… That is great! But you are not fully home yet.

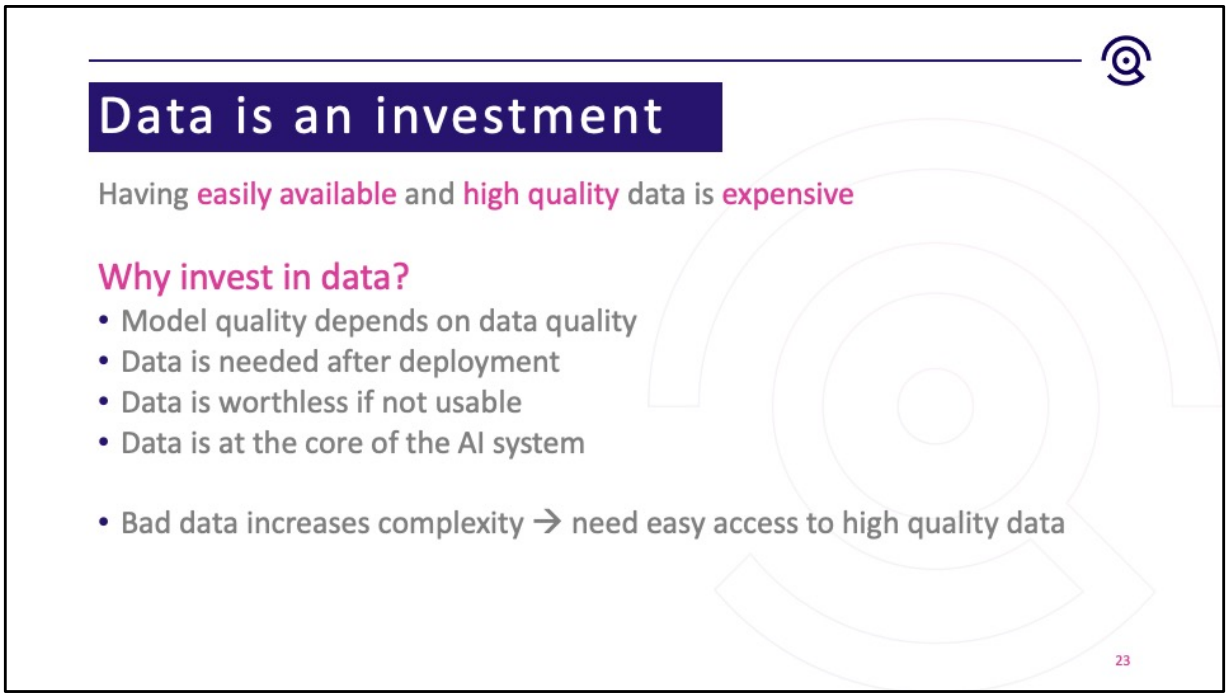
Let’s investigate some of the common deployment challenges that arise when bringing an ML solution to production.



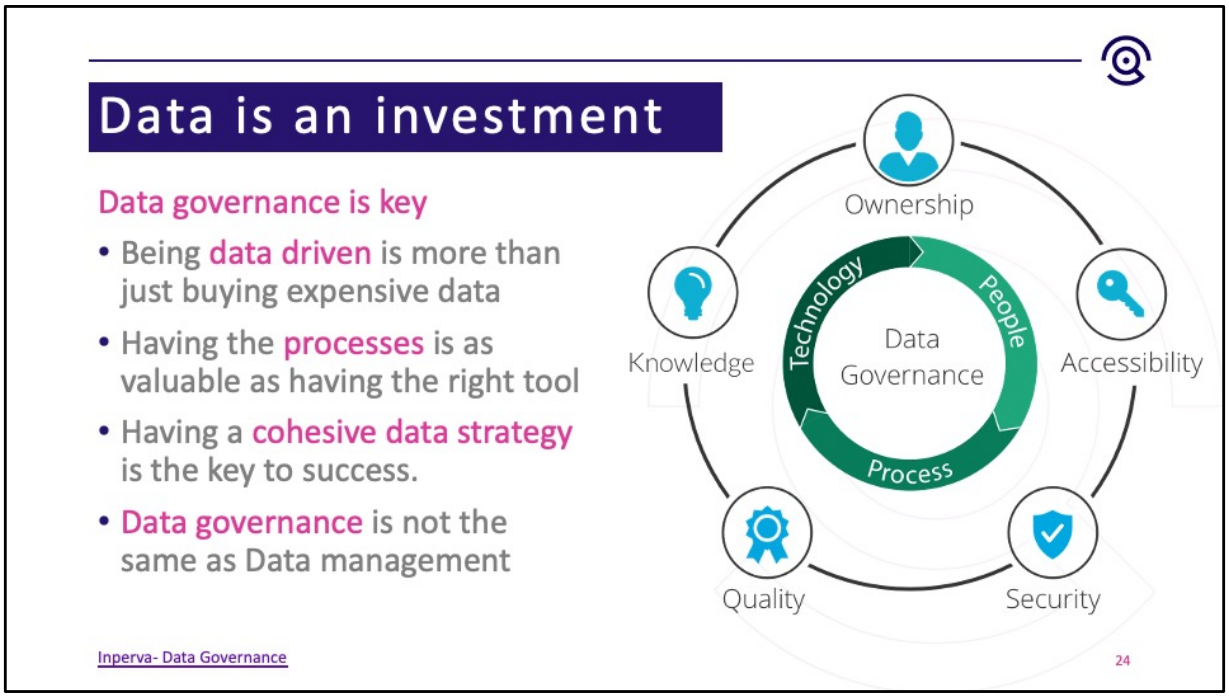
Building any software application comes with many challenges. However, new challenges start to arise when we also incorporate AI/ML in our solutions. Some examples are the following:

- Data is difficult to manage and resource consuming. Although we can extract immense value out of data, to be able to do so requires significant investment.
- Iteration is necessary but slow. Unlike in typical software engineering, getting feedback from making an update is not always immediate as it requires not only code changes but also data and or model changes.
- The expertise needed is abundant and diverse. The data scientist alone will not be able to understand the business environment as well as move the ML code to production (especially for more sophisticated projects).
- Scaling quickly becomes an issue. Scaling is extremely difficult, as it not only requires improving processes but also frameworks and infrastructure.
- Maintenance becomes particularly difficult, since the amount of things that need to be tracked in a simple workflow is already enormous.
- Selecting the right tool is not always so easy. AI related technology is abundant, fast-paced and ever changing. Is it almost impossible to keep track of all the changes in this landscape.

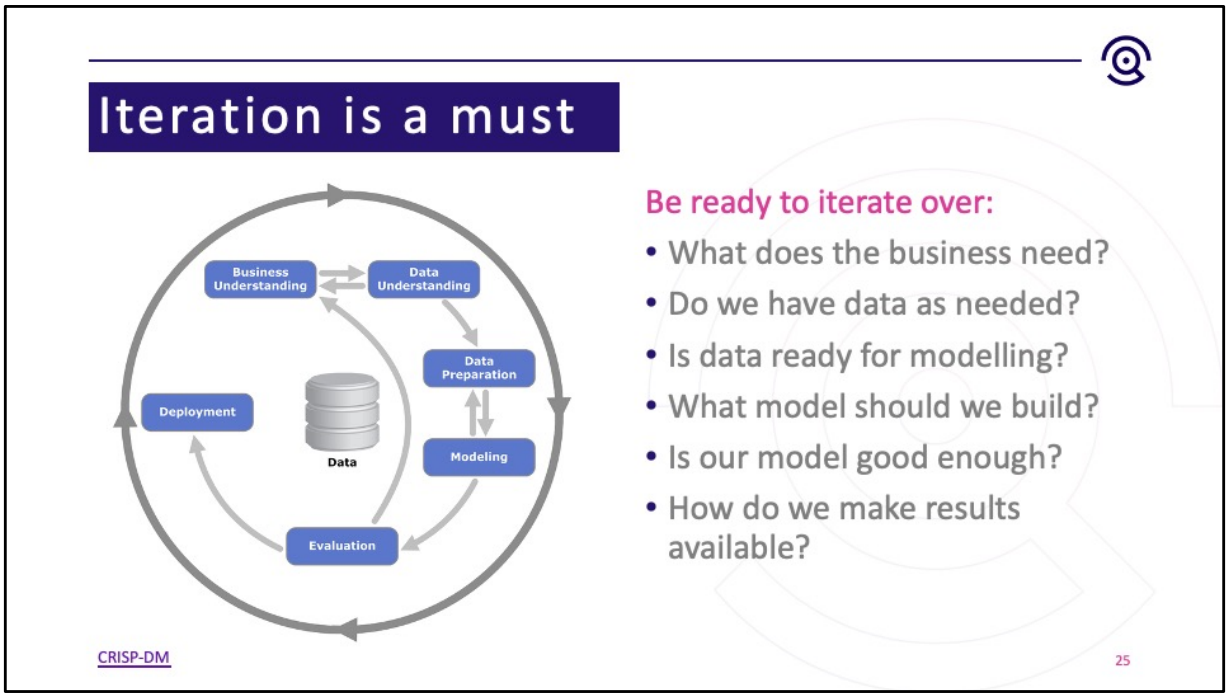
We will now dig deeper on each of these deployment challenges.



Being able to easily access and use high quality data on a continuous basis requires significant planning, infrastructure and resources. And this is of course not particularly cheap. However, it seems that everybody is currently focusing their strategies around being more data driven. Which poses the following question:
Why do we invest in data if it is so expensive?
In the first place, models need data, and the quality of those models is highly correlated with the quality of the data. So simply put, no data means no ML.
But even after the model has already been trained, that data might still be needed. So having our data in good shape facilitates many other process outside the ML system.
Moreover, it is common to hear that data is the new gold, oil or electricity. But if data is not usable then it becomes basically worthless.
Finally, when deploying ML systems, data is at the core of the AI system.
Poorly maintained data increases complexity, so high quality data needs to be easily accessible.



Investing in data goes beyond solely buying new technologies or expensive infrastructure, it also goes beyond buying expensive data sets or access to databases. Investing in data also means establishing solid processes around the data governance. Having good processes and best practices is as valuable as having the right technological tools.
A good data strategy is the key to success when it comes to maintainable AI solutions. It is important to notice that data governance is not simply the same as data management. Data governance links processes, people and technologies together, thereby allowing organizations to develop a cohesive data strategy.
For example:
- We need to establish who owns what data and also establish the protocols to access data and who can do so.
- We need to make sure data is secured and reliable and that the quality of the data is as high as it can get.
- Furthermore, all meta-data and additional knowledge coming from the data should be easily traced back to it.



In general, data science projects follow iterative incremental cycles before deployment makes results accessible to the stakeholders. This iteration normally goes over the following questions:

-What does the business need?

-Do we have the data as needed?

-Is the data ready for modelling?

-What model should we build?

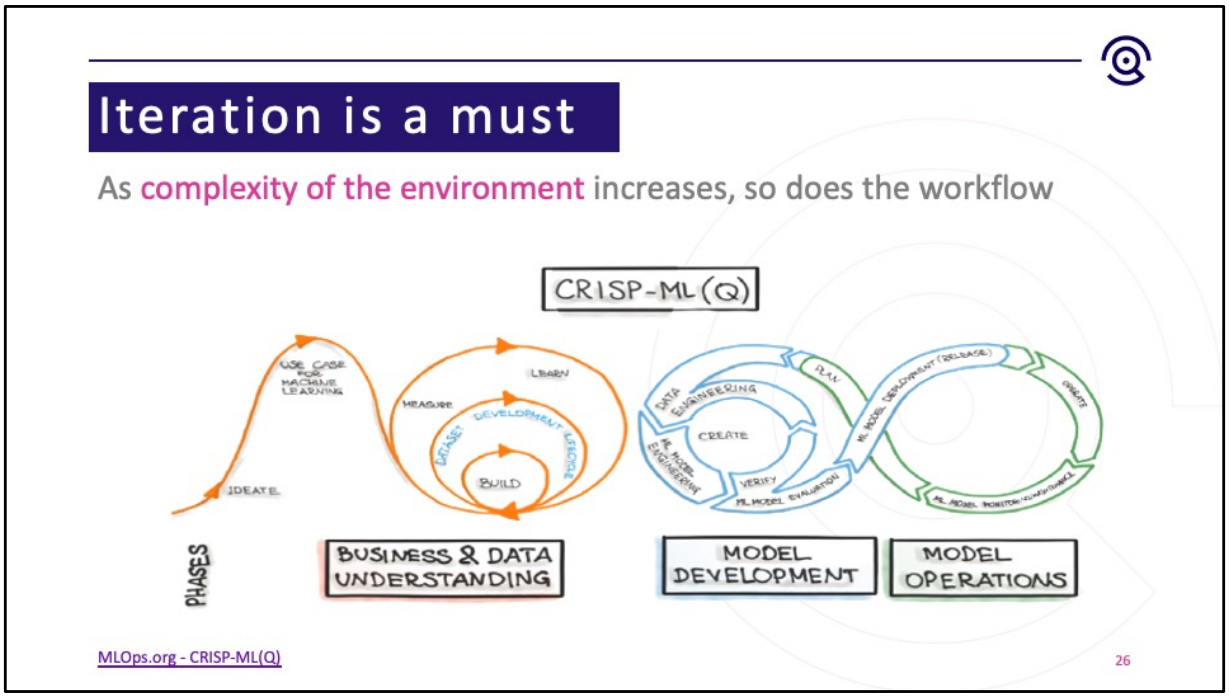
-Is our model good enough?

-How do we make results available?

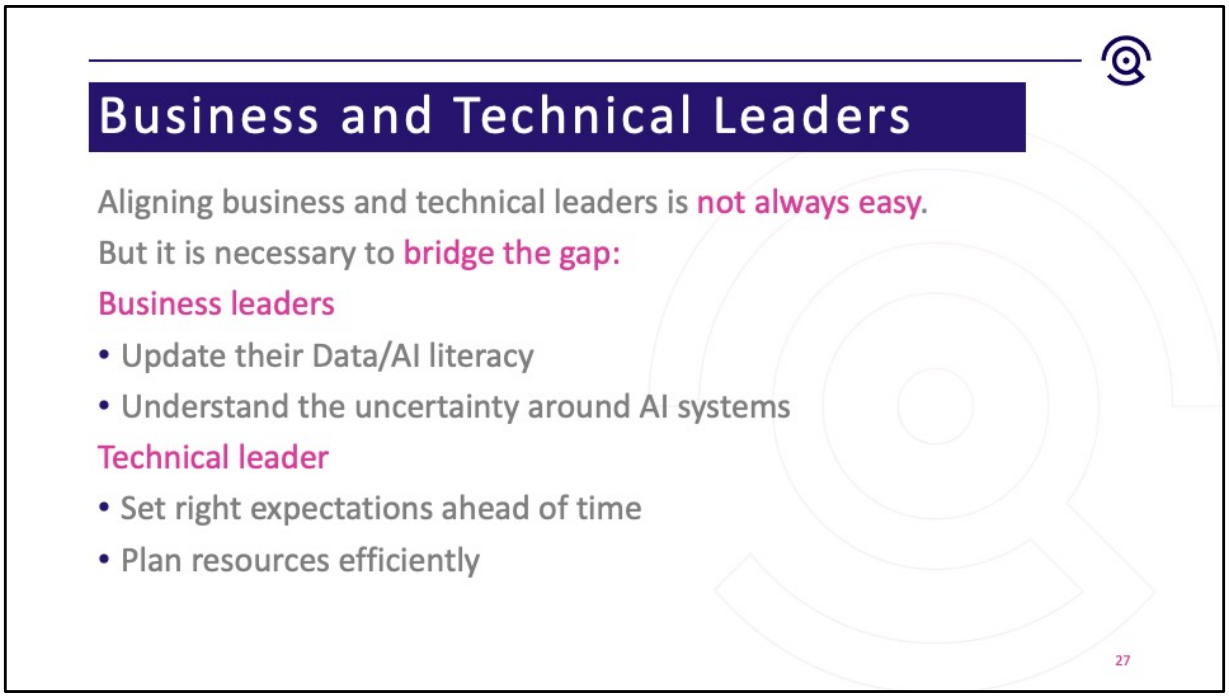
This iteration has been formalized since the early 90's, known as the CRISP-DM. This is a six face process model designed to capture the data science project lifecycle. It's like a set of guardrails to help data scientists plan, organize, and implement your data science (or machine learning) project. From an industrialization point of view, the major problem with this iterative process is that there is no guarantee that a satisfactory solution will be found. Data science solutions carry some inherent uncertainty and do not always guarantee to solve the problem at the expected level.

It is important to keep in mind that iterations can be slow and one iteration does not guarantee an incremental improvement. Therefore willingness to

iterate must prevail, as forcing developers to skip this process altogether will inevitably lead to poor results. Experimentation must be conscious and reproductible.



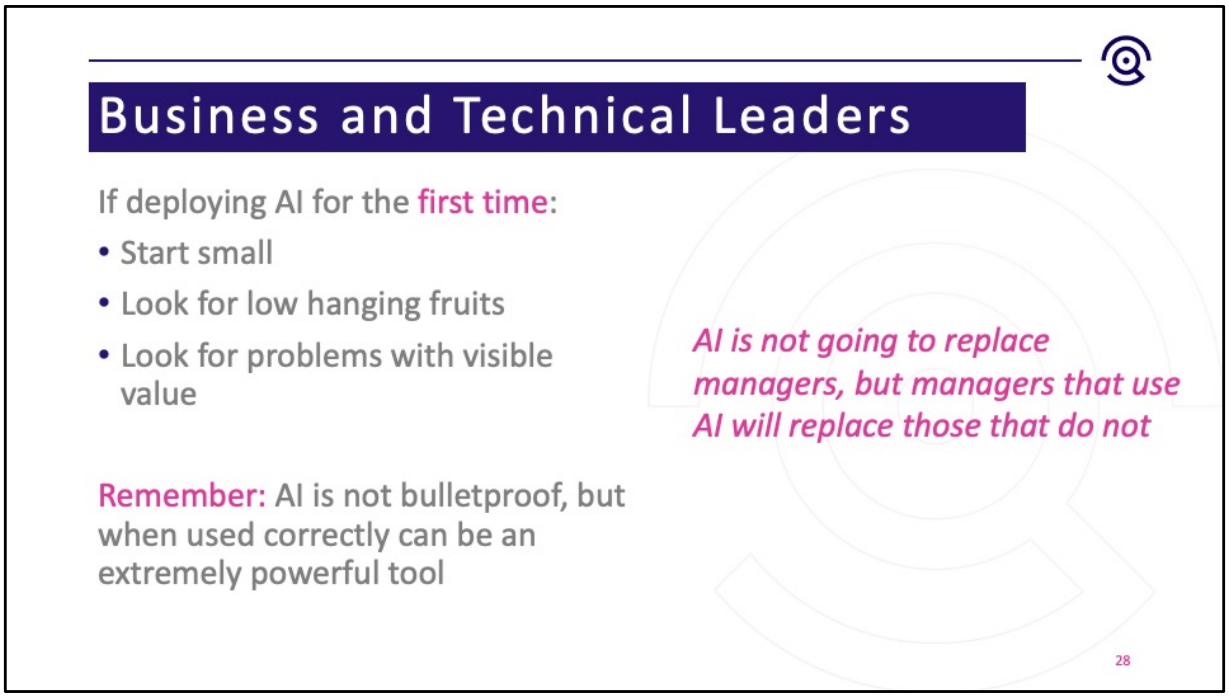
As the complexity of the environment and the solution increases, so does the complexity of the workflow and the iterative process. CRISP-MLQ improves upon CRISP-DM by adding the complexity of Data Engineering and DevOps to the picture. As AI systems evolve, and the solution becomes more general purpose, it is not only important to conceptualize the model development steps but also the operational side once it has been deployed to production.
Being able to continuously update our models and all other artifacts in production is a necessity for some AI applications. The other thing to bear in mind is that as the iterations become more complex, the time and resources requirements also increase. But it is still important that this iterative process is followed, otherwise it would be almost impossible to arrive to optimal solutions.



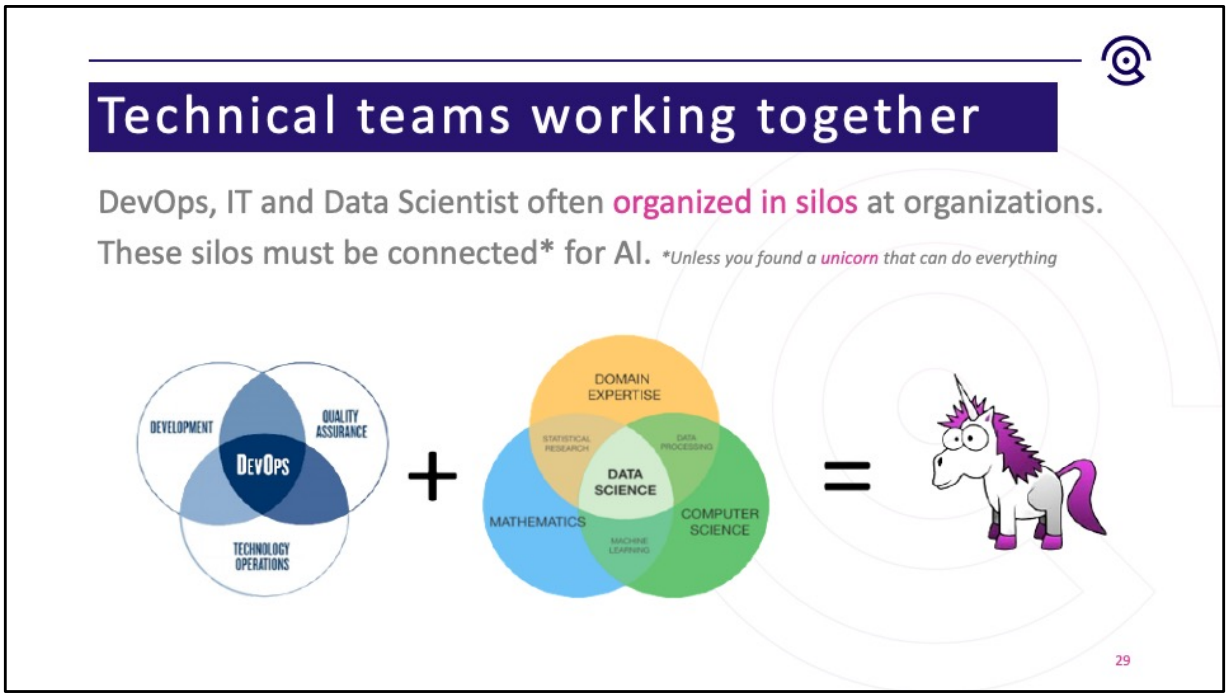
Aligning business and technical leaders is not always easy. Despite progress in recent years, there are still gaps in understanding between business leaders and technical leaders when it comes to deploying AI. Therefore it is extremely important to bridge the gap, and this is only possible if both sides are willing to reach out to each other.

It is important that Business leaders understand the possibilities but also the limitations of AI by updating their data and AI literacy. This will allow them to also understand the uncertainty around AI systems. After all, AI is not fool proof and it is very likely that mistake will be made while developing solutions.

On the other hand, as a Technical leader it is important to set expectations right ahead of time. Not only expectations about the performance, but about the process, as we just learned that we are dealing with an iterative process that does not always yields immediate results. Another important point is the resource planning, which should done very efficiently and take into account the iterative nature of the process.

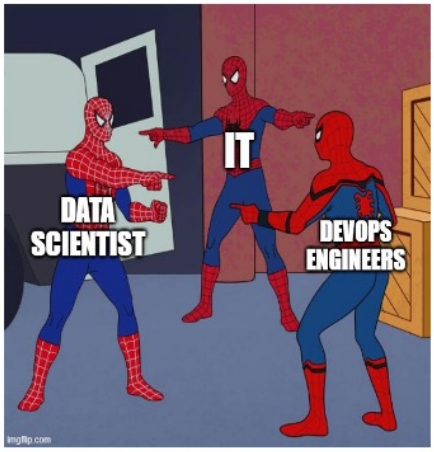


A very important remark is that in any organization, the most important thing should be generating value. So if you are deploying an AI model for the first time, it is important to try to select a business problem and AI solution where you can easily communicate its value to business stakeholders. A couple of tips & tricks in this regard are the following:
-Start small, there is no need to solve all business problems at once.
-Look for low hanging fruits, meaning that you should start with a problem that you are sure you can solve.
-And finally, look for problems with high impact where value is very visible.
Always remember that AI is not bulletproof, but when used correctly it can be an extremely powerful tool.
It is interesting to note that, as time evolves, the line between technical and business leaders in the AI field is blurring. And among most experts there is a consensus that AI is not going to replace managers, but managers that use AI will replace those that do not.



DevOps, IT and Data Scientist are often organized separately in silos at some organizations. Unless you found a unicorn that can do everything, these silos must be connected before pursuing AI.
Ideally your AI team would only be composed of these unicorns profiles that can do every step of the workflow. But the fact of the matter is that it is already very difficult to find good talent in any area of data science, software or data engineering and DevOps. This is the case because the demand for these profiles is currently very high. The main bottleneck is that although they are all technical profiles, the focus of each one is very different. For example, most of the time IT departments are not very familiar with the ins and outs of AI problems.
A good idea that has been adopted by large companies is the creation of cross functional teams that encompass various technical profiles in order to solve these challenges together. This of course only works if these teams have solid support from the organizations.

Technical teams working together

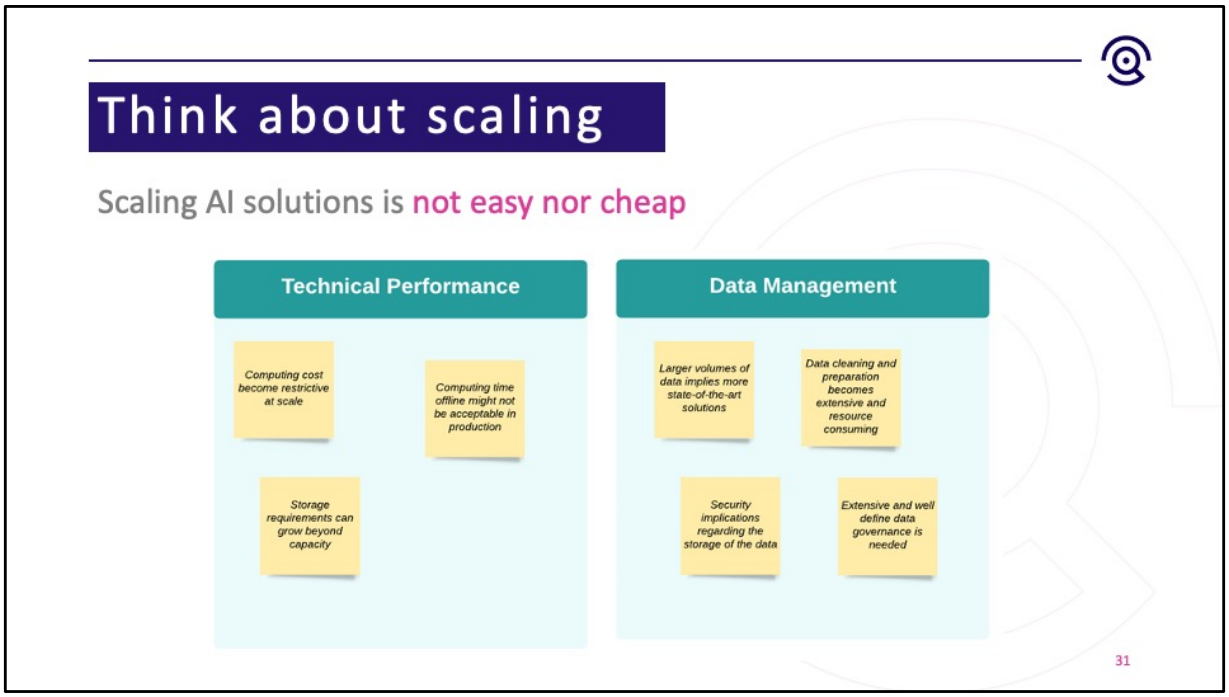


When working in silos:

- Impossible to have a high-level overview of the solution
- Constant blaming across teams
- Can't tackle complex problems (e.g. real time applications)
- Maintenance rapidly becomes a nightmare

30

Working in silos might have some benefits, but when it comes to AI there are some drawback that must be avoided:
- Since everybody is a specialist on a small part or some component, it is impossible for a single team or person to have a high-level overview of the solution.
- This leads to the constant blaming across teams when failures appears. In most cases these are just due to miscommunication or lack of understanding on the responsibilities of the other team.
- This generates massive overhead and makes it impossible to tackle more complex problems (e.g. real-time applications).
- Finally, if everybody is isolated, maintenance of your solution rapidly becomes a nightmare because is difficult to establish ownership.



One of the biggest challenges facing AI is scalability when going from the POC to production. We furthermore have to bear in mind that, in general, scaling solutions is not cheap either.

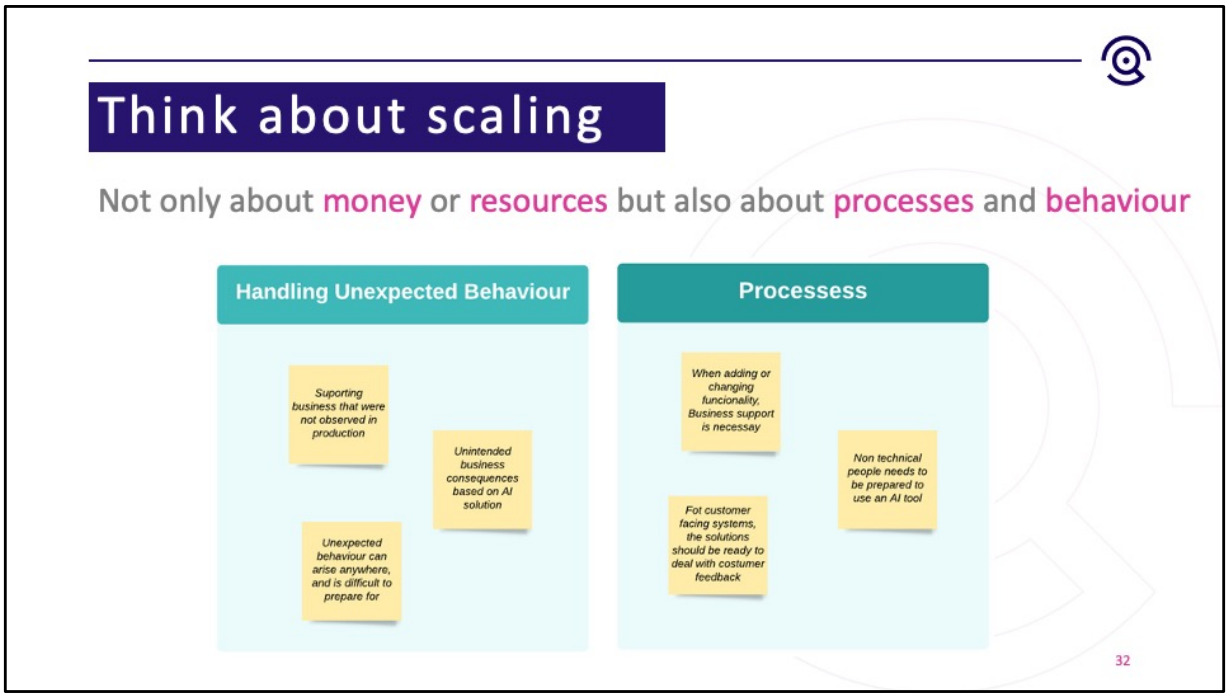
Technical performance of applications is always the first failure point when it comes to scaling. Data scientists at most companies are used to develop their solutions in notebooks that are set up in a different environment and often are not concerned with scalability problems. This drives them not to think about the difference in resources that a model consumes offline versus online. This leads to many problems like for example:

- The computing cost becoming too high and restrictive at scale.
- The computing time offline may not be acceptable in production (e.g. when a very good model simply takes too much time to give an answer).
- The storage requirements can grow beyond capacity.

Another limiting factor to scaling is the requirements for effective data management.

- Larger volumes of data require more expensive and state-of-the-art solutions.
- Also data cleaning and preparation becomes more complex and resource consuming in this case.

- New security implications regarding the storage of data can arise.
- And last but not least, extensive and well-defined data governance is needed. When building large applications, governance becomes more than just a regulatory necessity.



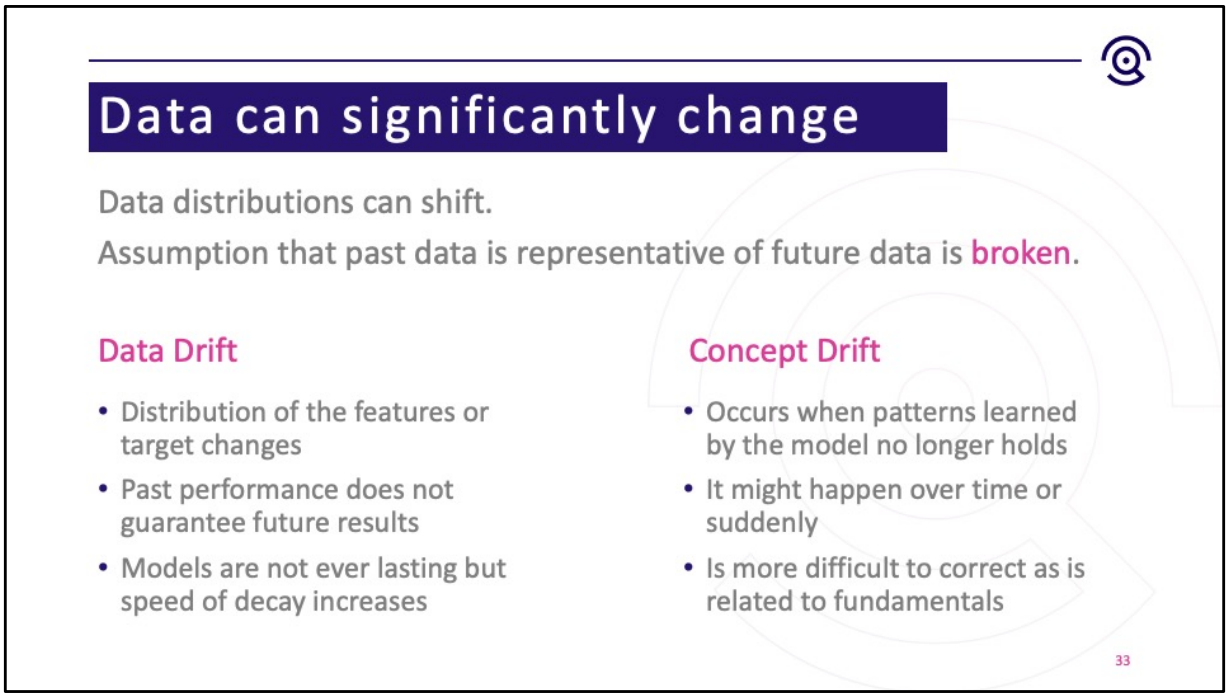
Scaling AI solutions is not only about money or resources but also about processes and behavior.

Handling unexpected behavior is a must. And the places from which this behavior can arise also grow. It is difficult to plan for this unexpected behavior, as in most cases they are unknown unknowns. But some case examples could be the following:

- Supporting business with behavior that was never observed in production.
- Or it could also be related to indirect effects, like for example unintended business consequences based on the AI solution.

Apart from unexpected behavior, it is necessary to prepare to deal with processes and people. There should be processes in place to deal with business feedback.

- Ideally, when adding or changing functionality, business support will be necessary as most likely these scaled solutions will be interacting with other processes.
- For this reason, the non-technical people should also be trained to use and understand these AI tools.
- Moreover, for applications facing the customer, it should be easy to deal with customer feedback.

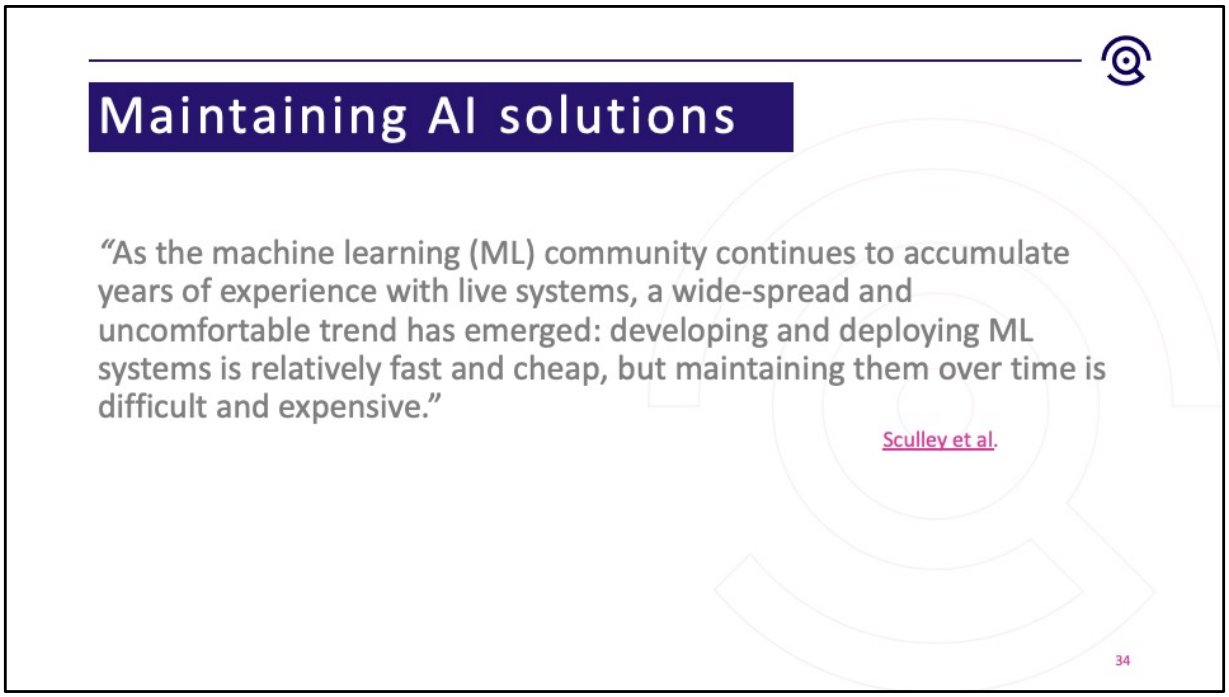


Another one of the challenges of deploying AI is that data distributions can shift. When this happens, the assumption that past data is representative for future data is broken. This means that the assumptions that hold our model together are violated as there is no point in trying to predict future behaviour if the past is not representative of the future. There are two main kind of drift or shift, namely data drift and concept drift.

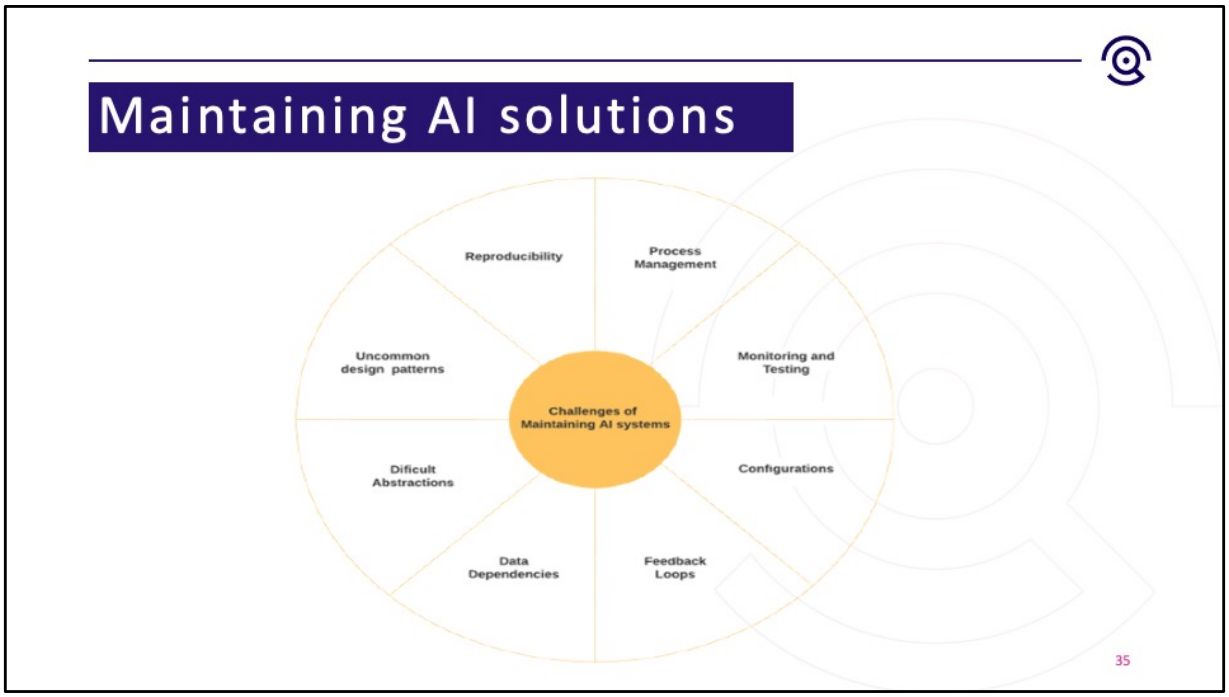
Data drift is encountered when the distribution of the features or target changes. This means that past performance does not guarantee future results. So even if the model used to be very good, that might not be the case moving forward. Although is true that ML Models are not ever lasting, in the presence of data shift the speed at which they decay increases.

Concept drift occurs when the patterns learned by the model no longer hold. This is driven by the change in the relationship between the predicted target and the data features used to predict the target. This might happen slowly over time or suddenly, for example after COVID the digital purchase behaviour of most of population has changed. The data that was used to predict such behaviour may no longer be a good predictor. The tricky part with this kind of shift is that it makes it more difficult to

correct as it is related to the fundamental relationship between the data. So in many cases completely new data would be needed.



Sculley et al. mention the following statement in their paper "Hidden technical debt in Machine Learning Systems": *"*As the machine learning (ML) community continues to accumulate years of experience with live systems, a wide-spread and uncomfortable trend has emerged: developing and deploying ML systems is relatively fast and cheap, but maintaining them over time is difficult and expensive."
This seems to be a bit counterintuitive, but perhaps the driving force behind this is that as some new tools make developing and deploying easier; many developers fail to take into account the cost of maintaining ill-designed solutions. Thereby making the hidden technical debt extend across solutions. In all fairness, in some cases this is all explained by the complexity of the new solutions itself, as the cost of building such solutions has decreased exponentially over time.



There are many dimensions to maintaining an AI solution. It is a topic so rich and deep that many books have been written on it. This picture summarizes some of the most interesting things that need to be taken care of to make maintainable AI possible. We have already been over some of these topics like:

- Process Management
- Configurations
- Data Dependencies
- Feedback Loops

We will dig deeper on the remaining of these topics when we discuss MLOps (or Machine Learning Operations) in the fifth module of this course. There we will even dedicate a full section to Monitoring and Testing.

Selecting right technology

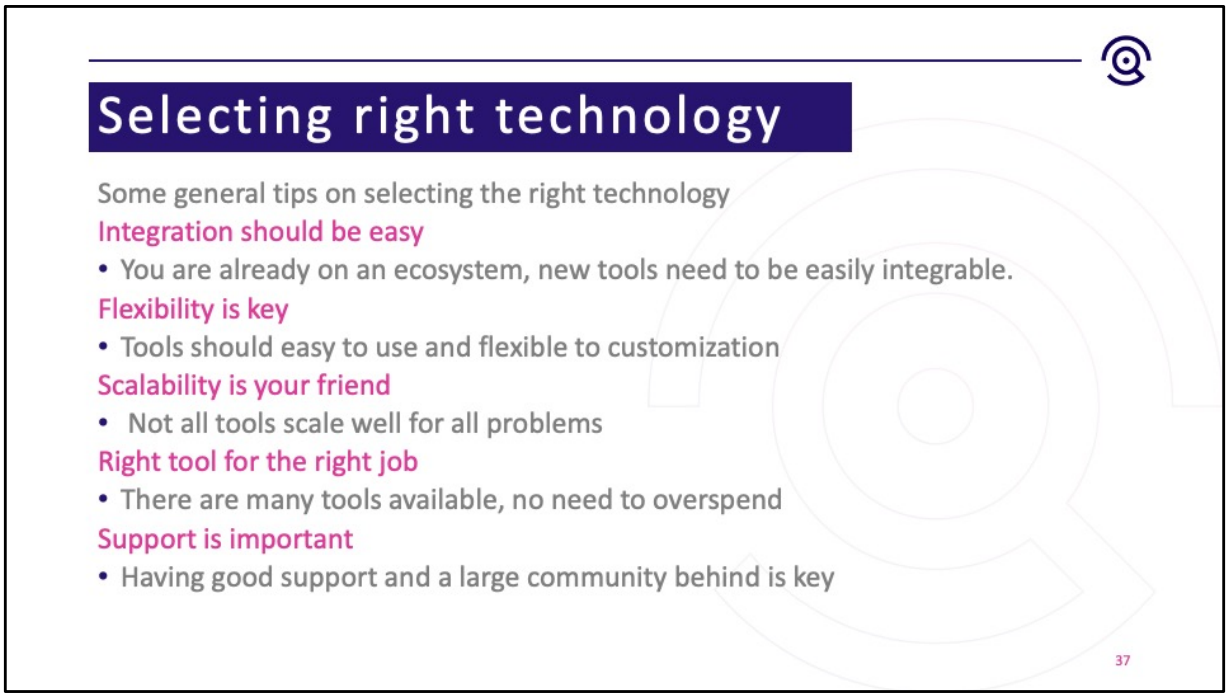
Selecting the right tool for the problem at hand is not always simple, as the technology supporting AI is

- Diverse
- Fast growing
- Tailored

Remember: Don't marry yourself to a tool. Tools are just means to an end.

36

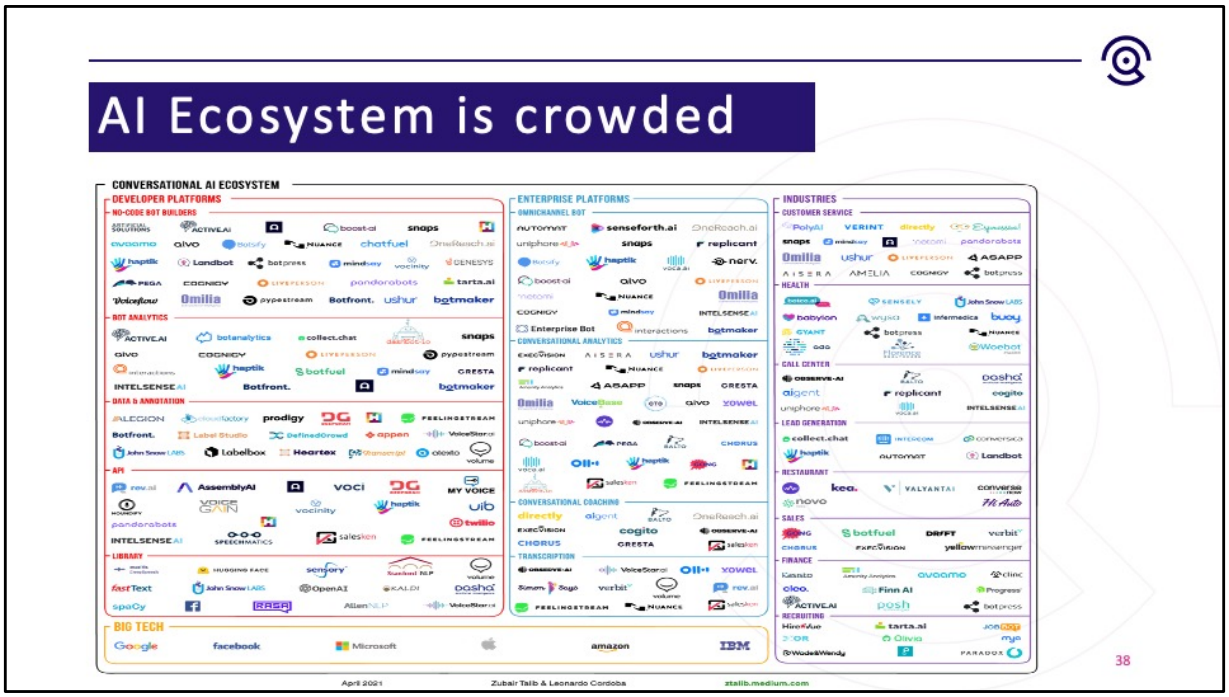
Selecting the right tool for the problem at hand is not always simple. The technology supporting AI is:
-Diverse and of different qualities and price tags,
-growing at a very rapid pace and
-generally tailored to solve very specific problems.
The main take away to remember is that you shouldn't marry yourself to a tool. Tools are just means to an end. It should be very easy to change between tools in most cases.



Some general tips on selecting the right technology are the following ones:

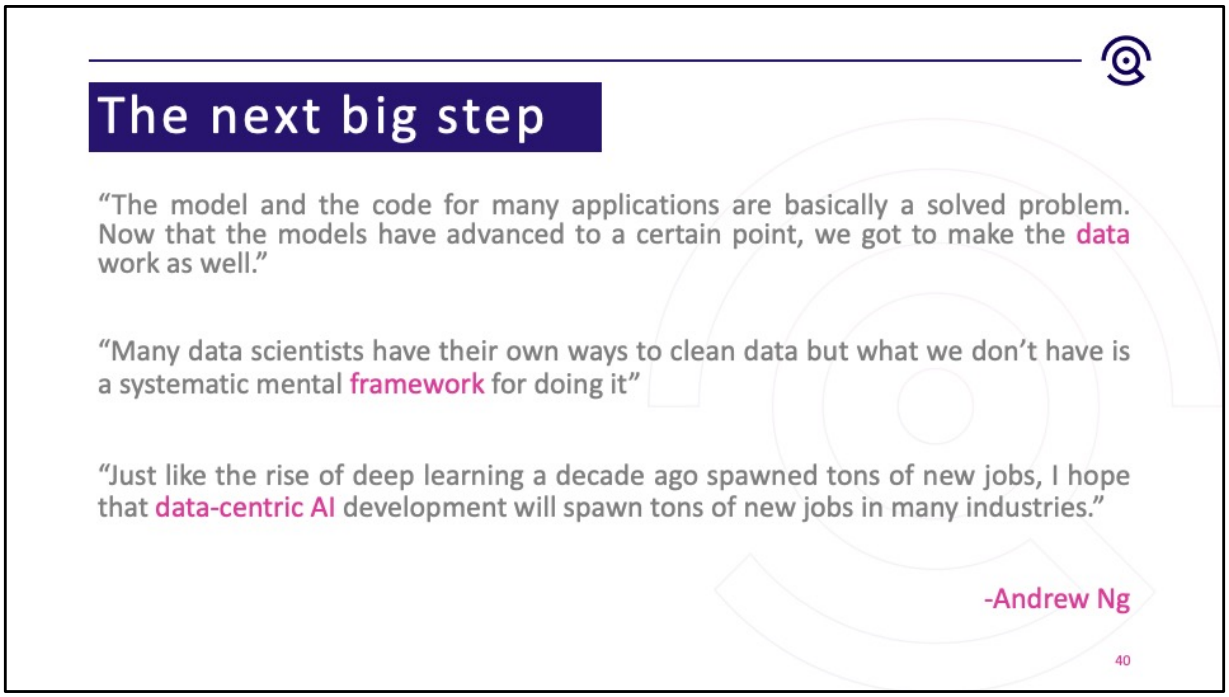
- Integration should be easy. When you are already on an ecosystem, new tools need to be easily integrated. There is no point in having to change your complete way of working just to change a tool. Unless it is a completely disruptive tool and a proper cost-benefit analysis has been made.
- Flexibility is key. Tools should be easy to use and flexible to customize. If your tools are too rigid or only work under very restrictive circumstances, perhaps it is a good moment to revaluate your choices.
- Scalability is your friend. Not all tools scale well for all problems. No matter what kind of tool you are choosing, always consider how easy would it be to scale to your current solutions with it. It might save you some head aches down the road.
- Right tool for the right job. There are many tools available, so there is no need to overspend. Choose your tools matching your solutions. There is never a one size fits all tool. Make sure you are choosing a tool that solves the problem at hand.
- Support is important. Having good support and a large community behind a tool is key. Although paid support is good, having a tool with a large community makes it more likely that other people have already gone through similar situations. If you don't have good support or a good community behind your tool choice, get ready for

a ton of time devoted to small bugs.



On last point to keep in mind is that the AI ecosystems are growing scaringly fast, which steadily increases the search cost of new tools. It is sufficient to see this ecosystem for conversational AI, which is just a small part of the capabilities of AI. The main message we want to convey is that you remember the tips we showed you for selecting tools when it comes to selecting your AI ecosystem. Do not marry yourself to a particular technology or vendor, as new and better solutions are released every day. Just make sure that you are flexible, cost effective and ready to evolve on a continuous basis.

Let’s now talk about data-centric AI and find out what it means.

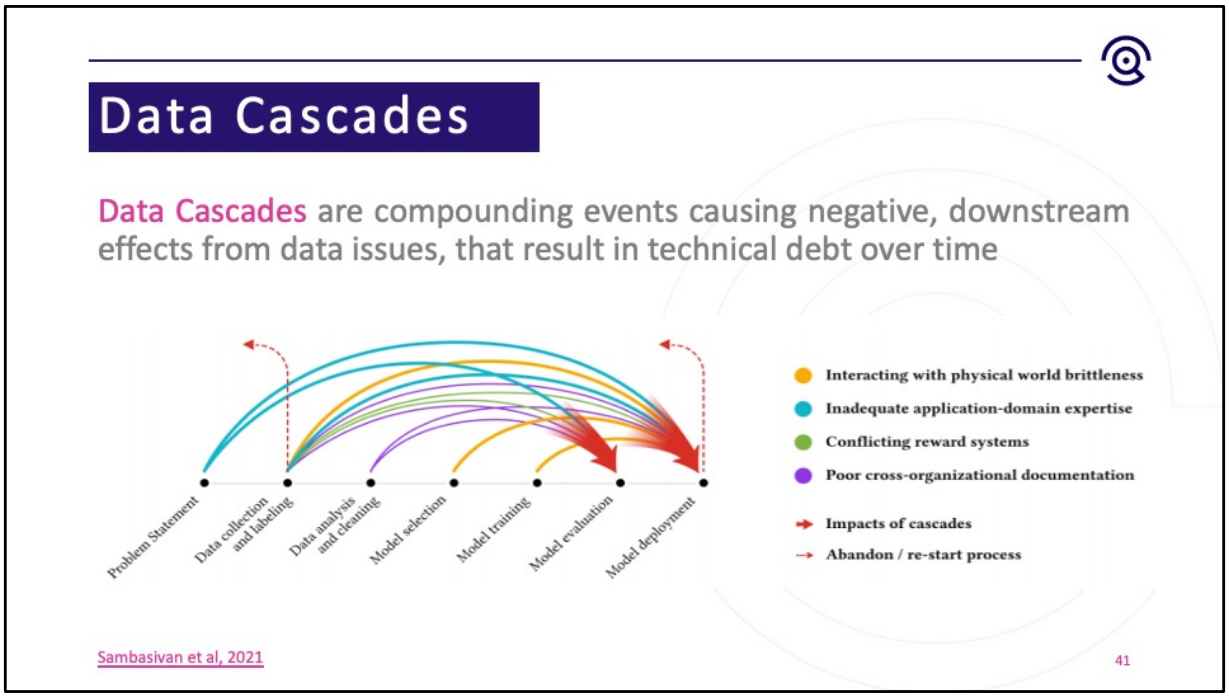


Big data and AI are not new anymore, we have learned how to deal with very sophisticated problems using advanced analytics and have even become very good at it. But can we do more?

To answer this question, let us take a step back. We have previously talked about the value of data in this course, and how data has become the new gold, oil or electricity. So why don't we improve the way we use it? Or is it even possible? Stanford Professor and founder of deeplearning.ai, Andrew Ng, has become a big advocate of this way of thinking. He states that: "The model and the code for many applications are basically a solved problem. Now that the models have advanced to a certain point, we got to make the data work as well.". This means that we should start creating a framework to systematically improve our data as we have already done with the models. Or as Andrew puts it "Many data scientists have their own ways to clean data, but what we don't have is a systematic mental framework for doing it.". This new framework is called data-centric AI and this change of mental framework has a huge potential economic benefit. We can see this in another statement of Professor Ng: "Just like the rise of deep learning a decade ago spawned tons of new jobs, I hope that data-centric AI development will spawn tons of new jobs in many industries."

Before we dig deeper on what being data-centric actually means, let us start with a

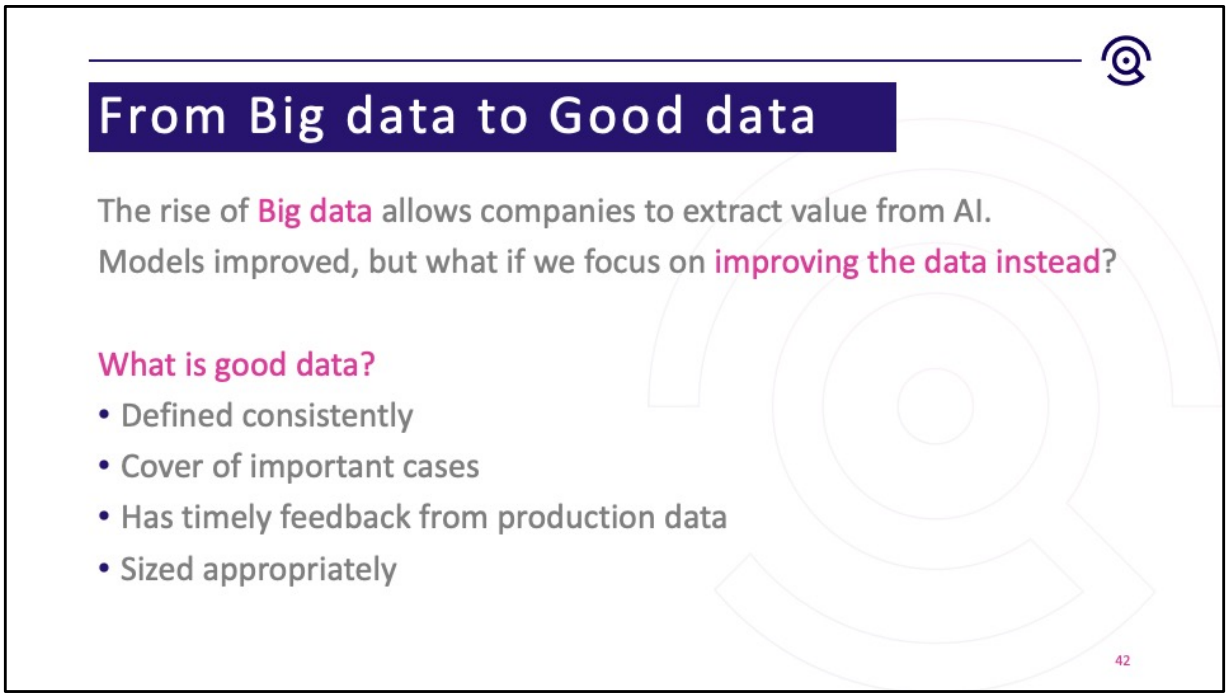
little motivation on why it is important.



Data is the central part of AI. Paradoxically, it is also the most undervalued and de-glamorized aspect of it. Many AI experts and practitioners tend to neglect the value of well-managed data. This often leads to what is known in the literature as data cascades. Following the same definition as in Sambasivan et al, 2021: Data Cascades are compounding events causing negative, downstream effects from data issues, that result in technical debt over time.

From the image we can see how these negative effects, arising from different places in the ML model workflow, can accumulate. In turn this is forcing either the re-start of the iterative process or even the complete abandonment of the project. These, sometimes seemingly small, errors or events can become exponentially expensive as the cascades propagate. Moreover, the effects of these cascades can manifest even a long time after ML systems have been deployed, which can make maintenance virtually impossible.

This creates the need for not only big data, but good data. So, the process of improving our data must be a systematic one. Moreover, if we could leverage this process beyond just cleaning and maintaining our data, we could potentially also improve our ML systems by means of just improving our data instead of improving the model or the code itself.



The rise of Big data has allowed companies to extract enormous value from AI systems. During this period, models have improved enormously. But what if we focus on improving the data instead?

What if instead of just focusing on exploding big data, we focus on having very good data to feed to our models?

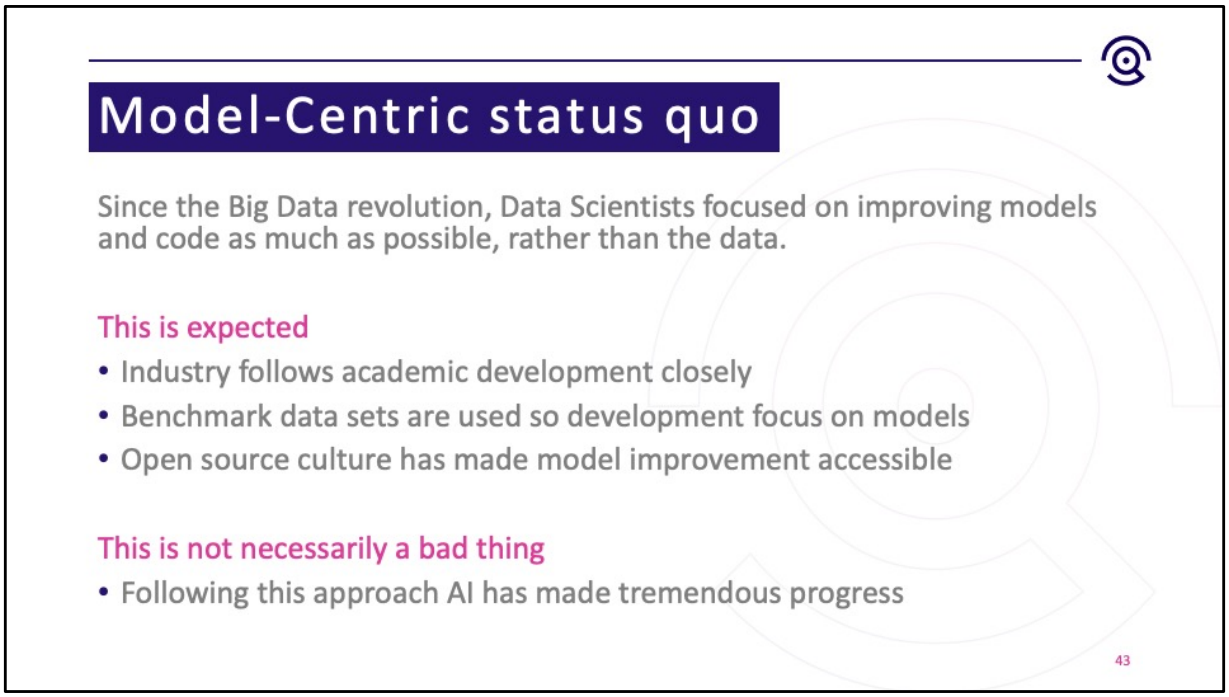
This leads to another question however, namely: What exactly is good data? Good data should have the following characteristics:

1-Good data should be defined consistently, for example that the definition of labels is unambiguous.

2-Good data should cover all the important cases, meaning that you need a representative set of input features in the data.

3-We would like timely feedback from production data such that the data distribution covers data drift and concept drift issues.

4- And good data should also be sized appropriately towards to the problems you are trying to solve and the ML models you are using to solve them.



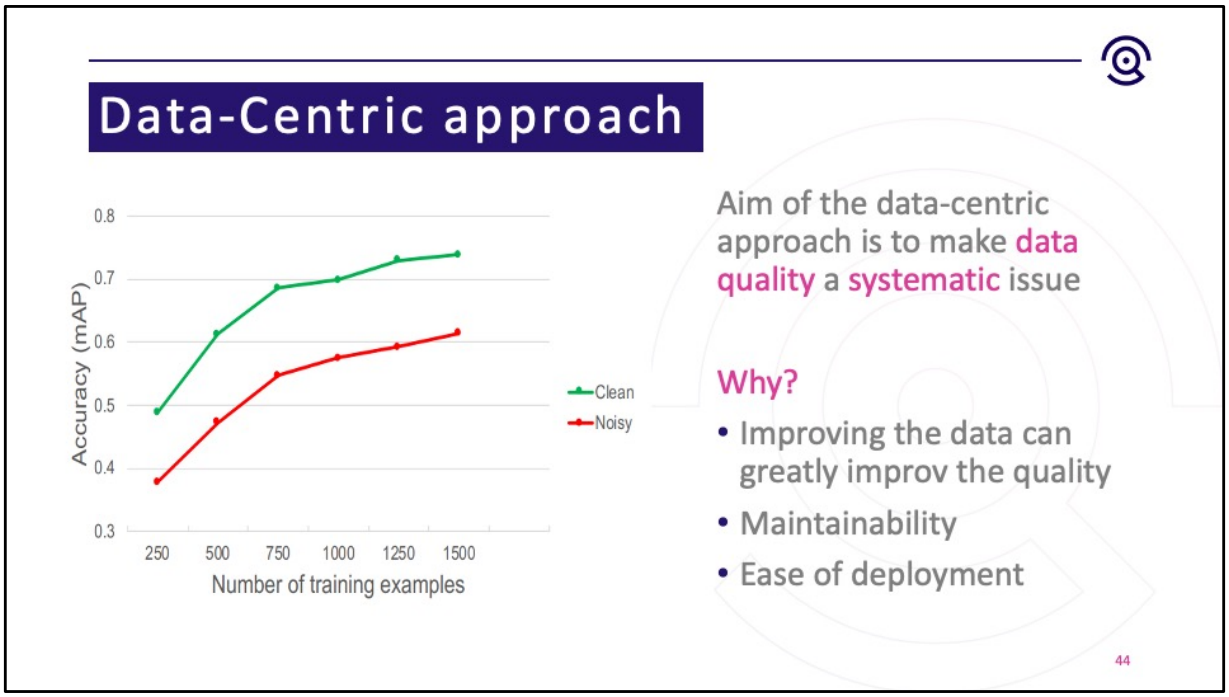
Since the Big Data revolution, and the explosion of advanced data analytics, data scientists have focused on improving the models and code as much as possible, rather than the data itself.

But this is not really a big surprise, it is actually expected to be the case if you think about it.

- In the first place, the industry follows academic development very closely. Most recent academic papers focus on improving the models, with some rough estimates indicating that as little as just 1% focuses on improving the data.
- The reason for this is that very often, benchmark data that have already been curated and properly tested are widespread used in certain domains. These benchmark datasets allow researchers to compare results, so naturally the development focuses mainly on how to improve the models.
- Another reason for this model-centric preference, is that the open-source culture has made model improvement accessible. It is very easy for somebody with access to GitHub to just grab a benchmark model and try to improve on it by fine-tuning it towards to the problem at hand.

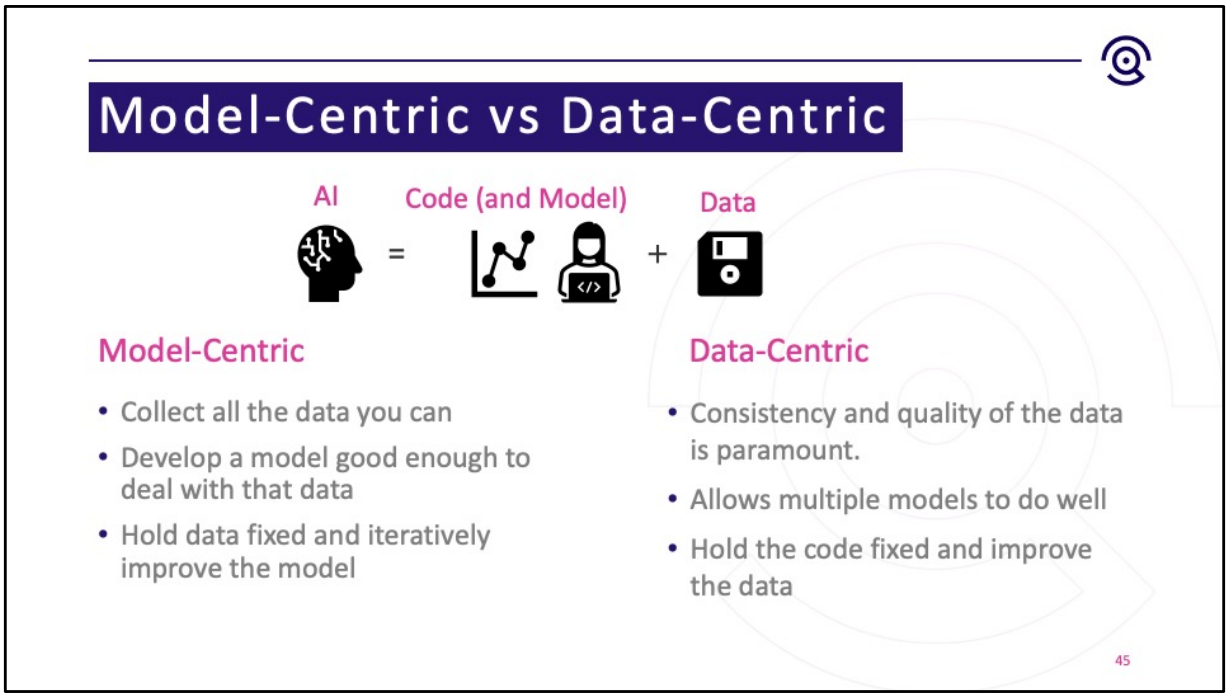
And in all fairness, all of this is not necessarily or even at all a bad thing. Following this approach towards AI has made tremendous progress in the last years. But now that the models have advanced to a certain point, we have the need for the data to

catch up as well.

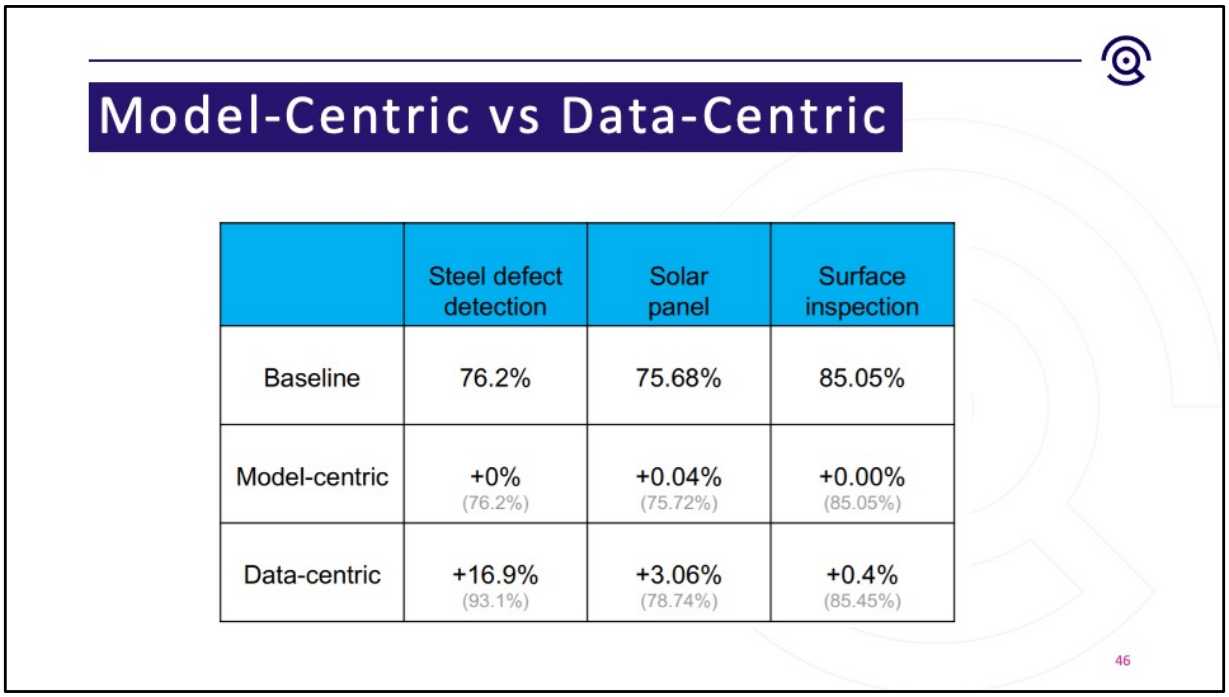


So how can we make the best use of the data? How can we change the mental framework and become more data-centric? The Aim of the data-centric approach is to make data quality a systematic issue.
We have to keep in mind that being data-centric is not just related to having the right tools for it, nor a particular algorithm nor a one-size fits them all kind of solution. It is about creating neat processes to govern the mental framework, so the data is consistent and of high quality. And the changes we make to improve that consistency have to be systematic. That way we can hold our models fixed and evaluate the impact that improvements on the data make on the model's performance.
But why should we change? Here are a couple of reasons to do so:
- Improving the data can greatly improve the quality of the whole AI system
- Maintainability becomes easier, and like we have seen in this course before, this tends to be one of the most difficult things to do properly.
- Additionally, good clean data eases the whole deployment process.
On a reaffirming note, we can see in the picture that improving the data quality outperforms increasing the sample size. Take for example the noisy dataset with 500 samples. Tripling the dataset's size gives about the same performance gain as simply using a clean dataset of 500 samples. So no big data, but good data. This is especially helpful when dealing with smaller sample sizes. And there is a big economic incentive

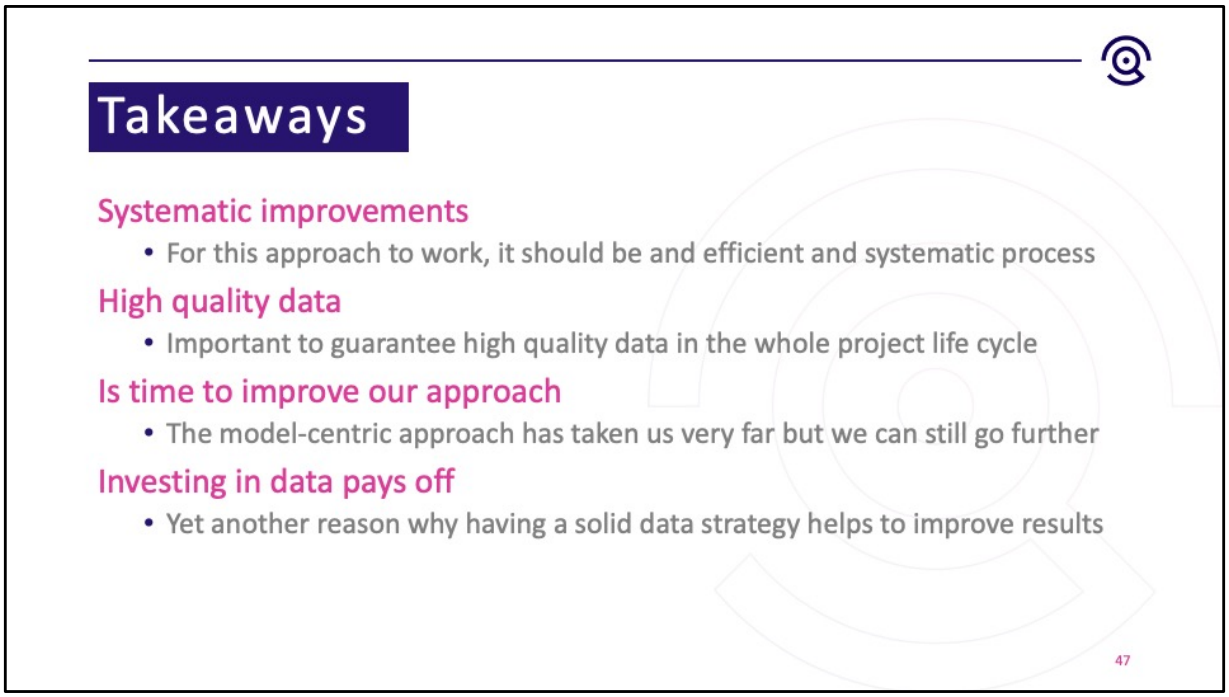
here, because there are many economic sectors that are still not using AI solutions because of the lack of data. However, imagine that we can generate good results in smaller samples by means of just improving the quality of the data. Well, then we could bring AI to places were there is still a lot of unexploited potential.



Now lets us compare the two frameworks. As we have discussed before, we can abstract an AI system as the sum of 2 components: the Code (or more accurate, the software code and ML model) and the data. Being model-centric or being data-centric is just a matter of which component we choose to optimize.
In the model-centric approach we collect all the data we can and develop a model good enough to deal with that data. We therefore hold the data fixed and iteratively improve the model.
The data-centric approach does basically the same but around the data. Here, the consistency of the data is paramount. We need to use the right tools to improve the data quality. This way we'll allow multiple models to do well because of the data. And unlike in the model-centric approach, this time we hold the code fixed and improve the data iteratively.
Please note that in both cases we work iteratively to achieve better results. It is very important that, as a manager, you understand that this part of the process is regardless of the framework. Also understand the value of not rushing this process just to get results faster, because more likely you are just pushing the problems forward to when it is time to maintain applications in production.



Here we show three examples with some performance measurements for steel defects, solar panels and surface inspection applications. The baselines contain benchmark models that are develop properly let's say. As we can see, in all three cases, the model-centric approach is not showing any improvements over the baseline. The data-centric approach on the other hand is showing some improvements. Not always, but sometimes a little bit and sometimes even a lot. This is just to show how we can still make some performance gains if we focus on the data after we have exhausted the possible improvements via the model.
This does not mean that improving the model is not important, but once you have exhausted the possibilities of improving the model, there is always place to improve the data.



Now let's go over the main takeaways for data-centric AI:
- We need systematic improvements. For this approach to work, it should be an efficient and systematic process. Remember that it is about the framework, not the tooling.
- We focus on high quality data. It is important to guarantee high quality data throughout the whole project life cycle. Better data means better systems.
- It is time to improve our approach. The model-centric approach has taken us very far, but we can still go further. There is lot that can be gained by just improving the way we think about our data.
- And finally, investing in data pays off. This is yet another reason why having a solid data strategy helps to improve business results. Remember, that investing in data is not only about tooling and databases, but investing in data is about processes, frameworks and having the right mind set.

A new systematic framework

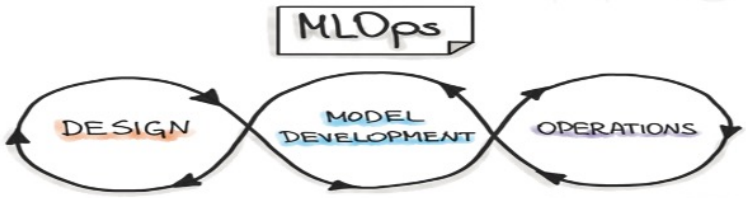
We know

Making data quality systematic improves general performance

But…What if?

We use the same principle for the whole project lifecycle

Enters…



48

As we have seen in this section, we can improve the performance of the whole AI system, if we are willing to change our mental framework to be more systematic when it comes to data. But then we can ask ourselves: What if we use the same principle for the whole project lifecycle?

Here is the entrance of Machine Learning Operations, most popularly known as MLOPs. This is a framework that establishes a general set of guiding principles to do just that. Bringing together the design, model development and operations under the same roof.

In the fifth and final module we are going to expand on this useful framework, where you will be able to understand exactly what it is and why it is so useful when it comes to the implementation of AI systems.

This is the end of the fourth module. I hope to see you again in the next and final module where we will discuss how to monitor deployed AI solutions, bye.