# **How to Manage Machine Learning Products**

## Part I: Why is managing machine learning products so hard? And why should you care?

Bastiane Huang

#### [Bastiane Huang](https://medium.com/@Bastiane?source=post_page-----386e7011258a--------------------------------)

#### [Sep 30, 2019](https://towardsdatascience.com/how-to-manage-machine-learning-products-part-1-386e7011258a?source=post_page-----386e7011258a--------------------------------)

## Summary: here’s what I want you to remember about this series of articles:

1. Managing ML products is more challenging than managing normal software products because it involves more uncertainties and requires not only technical but also organizational changes.
2. ML is best suited for making decisions or predictions.
3. Clearly define the problem, scope out requirements, set the metrics, and give engineers and scientists enough space and flexibility to explore before deciding on the path going forward.
4. Think about your data strategy from day one.
5. Building ML products is interdisciplinary. Think beyond ML.

Image for post



In my [previous article](https://medium.com/swlh/redefining-robots-demystify-next-generation-ai-enabled-robotics-fec64bfeb66c), I talked about the biggest difference that Machine Learning (ML) brings: **ML enables a move away from having to program the machine to true autonomy (self-learned)**. Machines make predictions and improve insights based on patterns they identify in data without humans explicitly telling them what to do. That’s why ML is particularly useful for challenging problems that are difficult for people to explain to machines. It also means that ML can make your products more personalized, more automated, and more precise. Advanced algorithms, massive data, and cheap hardware are enabling ML to become the main driver of GDP.

The adoption of ML has been rapidly advancing across various business sectors. Nearly half of the companies have incorporated one or more artificial intelligence capabilities in their process and another 30% are piloting AI projects, according to [Mckinsey](https://www.mckinsey.com/featured-insights/artificial-intelligence/ai-adoption-advances-but-foundational-barriers-remain)’s recent survey. It’s not hard to see why ML is expected to be even more transformative than mobile technology. However, the transition to ML could also be more than 10 times harder than the transition to mobile. Before we talk about why that’s the case, let’s go through the basics.

# Machine Learning Basics

## What are AI and ML?

There’s no universally agreed definition of AI and the definition changes all the time. Once a certain task is performed by a machine, the task is no longer in the scope of AI. ML is a subset of AI. CMU professor Tom M. Mitchell defined Machine Learning to be a study of computer algorithms that allow computer programs to automatically improve through experience.

## Types of ML

There are three main types of machine learning :

* **supervised learning**: The most common one and widely used type of learning. The algorithms learn from labeled data, i.e. training data sets that are tagged with the outcome the model is trying to predict. In short, it’s about predicting outcomes.
* **unsupervised learning**: On the other hand, unsupervised learning algorithms learn to identify patterns in the data without labeled data. It can be used in clustering, association, and anomaly detection problems. There’s also semi-supervised learning which is essentially a hybrid between supervised and unsupervised learning.
* **reinforcement learning**: The algorithms learn as they get feedback on corresponding predictions over time. RL is used in control domains such as robotics or self-driving cars.

## Types of Machine Learning Products

Depending on the types of products and where the core values come from, you will require different skill sets in your team and need to focus on different parts of the products.

* **Enterprise vs. Consumer**

Consumer ML products such as smart speakers have a stronger social component than their counterparts in enterprise segments. Therefore, user experience (UX) plays a more critical part in designing consumer ML products and ML tends to become an enabler for better UX. For example, NLP (natural language processing) is used to improve the interaction between Alexa and its users. On the other hand, the core value of enterprise, especially industrial ML products, such as predictive maintenance software, tends to come from the functional performance (e.g. accuracy) of their predictions. This is not to say that UX is not important for enterprise ML products. However, this is something to consider when you only have limited resources and need to focus on optimizing parts of your products.

* **Are you building an ML product or applying ML to your product?**

If the core value of your product comes from ML models, then you are likely building an ML product. On the other hand, if ML is only used to enhance the experience or performance of your product, then you are most likely applying ML to your product. In this case, it’s essential to understand the input and output of the models but not the technical details like architecture or whether the ML models are based on CNN (Convolutional Neural Network) or R-CNN. For example, the model takes demographic data of users to predict their monthly spending on the platform. Many companies or teams will also leverage existing solutions so they don’t reinvent the wheel. On the other hand, building ML products often requires PMs to be more technical to help the team navigate key decisions and trade-offs.

The organization structures also vary. For companies building ML products or large corporations with heavy investments in ML, like Facebook and Google, it’s common to hire ML researchers/scientists and pair them with ML engineers. On the other hand, for companies applying ML to their products or smaller companies with resource constraints, it’s probably better to hire multi-disciplinary ML engineers or train your software engineers to learn ML instead of hiring ML researchers/scientists.

* **Building ML products is often interdisciplinary.**

Even if you are building an ML product, it’s rarely the case that it will only involve ML. It’s often interdisciplinary and involves not only ML models but also software engineering, back-end infrastructure, data analytics, UX/UI design, and sometimes hardware. PMs need to be able to manage cross-functional teams and deal with interdependencies and potential clashes among teams. ML is fundamentally different from other disciplines as we will explain more in the following paragraph. It becomes even more complex if you are building ML products for the physical world like robotics or self-driving cars. PMs need to know what can and cannot be done with ML and when we should and should not use ML.

## Other key ML concepts to understand

* **Overfitting**: is a type of error that happens when models are too closely fit a specific set of data points. Robust ML models will perform well not only on “training datasets” but also on “validation datasets”. However, in the case of overfitting, the performance on the training data increases but the performance on unseen (validation) data becomes worse.
* **Deep Learning (DL)**: primarily used for image classification. DL uses a deep neural network and takes labeled images as input. Each layer of the neural network will transform the input into a slightly more abstract and composite representation. Eventually, the model learns to recognize objects in the images.
* **Natural Language Processing (NLP)**: a field of computer science for machines to understand human languages. It doesn’t necessarily involve ML. NLP is used for chatbots, voice assistants, or preprocessing data.

# Challenges in Managing ML Products

## 1. Experimentation is a crucial part of ML.

Just because ML involves code and data doesn’t make it similar to software engineering. In fact, the two disciplines couldn’t be more different. Unlike software engineering, developing machine learning products takes a lot more experiments and therefore involves more uncertainties and variabilities. Software engineering is a deterministic process of writing rules for machines to follow while machine learning is more probabilistic as it automates the task of writing the rules.

For example, if you want to teach a machine to recognize a cat. With software engineering, you may come up with rules like “a cat has 4 legs and 2 pointy ears.” But how is that different from a dog? If you use deep learning, instead of explicit rules, you will feed the machine with a bunch of cat photos (labeled images) and let the machine learn by itself. By doing so, you let machines write the rules by themselves. What you and your team do is to define the problem, prepare data, build a set of models, test, and iterate until you have a model that delivers desired results.

That’s why teams generally need to take more risks when developing ML products. It’s important for PMs to help set the right expectations to avoid potential clashes among teams. For instance, software engineers may feel that ML team is not giving them clear enough requirements without appreciating the nature of ML products. It’s also crucial to have engineers work closely with researchers/scientists so they can balance each other. More importantly, it’s better to have end-to-end systems working sooner to make sure that the algorithms that ML teams have been working on actually aligned with business goals.

## 2. Developing ML is a highly iterative process.

As mentioned before, ML is well suited for solving problems that are too complicated for humans to program explicitly. Models need to be trained, tested, and tuned. Often times scientists have to test a few approaches before choosing a satisfying one. That’s why it’s often more difficult to define milestones and estimate the timeline for ML products. Due to the nature of ML products, it’s critical for product managers to clearly define requirements and metrics and ensure that the team frequently test models against desired metrics.

## 3. There are more structural challenges beyond technical ones.

Because ML is so different from software engineering, it requires some fundamental organizational changes: experimental culture, data analytics-driven mindset, and more openness towards uncertainties, to name a few. Incumbents could face “innovator’s dilemma” if they treat ML as a purely technical problem and overlook the associated organizational changes. It is particularly challenging for companies such as robot makers who used to pursue high precision to develop ML products internally. In addition, ML products need large datasets for training. Companies need to build their own data pipeline and infrastructure to support the scaling of ML products.

## 4. ML is still a new field and it will keep evolving.

The term “software engineering” first appeared in 1965, 15 years after programming languages started to appear. Almost 20 years later, the Software Engineering Institute was established to manage the software engineering process. And today we have generally accepted best practices for software engineering. Machine learning, on the other hand, only started to flourish as a separate field in the 1990s. Deep learning, a subset of ML that has set new records in accuracy for many problems including image recognition and NLP, wasn’t widely discussed until the rise of AlexNet in 2012. Compared to software engineering, ML is still in its infancy and therefore lacks industry standards, metrics, infrastructure, and tools. Companies are still exploring best practices and kill applications.

## 5. Explainability and interpretability issues with ML products.

Many ML algorithms lack transparency, acting like a black box that takes input (e.g. images) and outputs predictions (e.g. what/who the objects/people in the images are). This makes it difficult for product managers to explain how ML models work and get buy-ins from users and stakeholders. Especially in critical domains like healthcare, accountability and transparency are extremely important. It’s challenging to ensure alignment between ML work and customer problems without a clear understanding of how an algorithm actually works.

With all these challenges, how should we go about managing ML products? Where do good PM instincts go bad for ML products? In Part II, I will talk more about my learnings and best practices.

## Connect Deeper: If you enjoy reading this article, please [subscribe to my personal blog here!](https://bastiane.substack.com/)

## **[How to Manage Machine Learning Products — Part II](https://towardsdatascience.com/how-to-manage-machine-learning-products-part-ii-3bdabf91eae4)**

### [Best practices and things I’ve learned along the way.](https://towardsdatascience.com/how-to-manage-machine-learning-products-part-ii-3bdabf91eae4)

#### [towardsdatascience.com](https://towardsdatascience.com/how-to-manage-machine-learning-products-part-ii-3bdabf91eae4)

*Bastiane Huang is a Product Manager at OSARO, a San Francisco based startup building software-defined robotics. She has worked for Amazon in its Alexa group and with Harvard Business Review as well as the university’s Future of Work Initiative.* [*She writes about ML, robotics, and product management. Follow her here*](https://bastiane.substack.com/).

*This post has been published on* [*www.productschool.com*](http://www.productschool.com/?fbclid=IwAR2D6uU6IAZmeNGLtBWs2POMn-pk9i8C7O4u24emInePlGJSyYIvgR7Mhfk) *communities.*