How to successfully manage your machine learning product

Suvrat Hiran Follow
Jul 13 · 6 min read

This article was first published in in Freshworks.

Building a software or product feature can be challenging in the best of times, even with defined workflows involving an army of developers. Now, with the pandemic compelling companies to make rapid advancements in software development and automation, project management can seem daunting.

Imagine having to develop a machine learning product or feature in the midst of a lockdown. If distributed teams are to remain a prominent workplace feature, and offices are to be reimagined, companies need to re-orient themselves for complex projects factoring in the new realities.

Last year, I had the opportunity to lead a machine learning project to build a smart de-duplication tool that would remove duplicate and incorrect data that clog databases everywhere. The tool, which needed to be deployed at scale, was among the earliest projects to be built by a Freshworks team based outside the company's office in Chennai, India. I learned a lot from that exercise.

In this blog, I delve particularly into developing a machine learning feature for a product company. Say, a customer churn prediction feature for a telecom company based on call activity, or lead scoring for a maker of customer relationship management products. On the consumer-facing side, think of a newsfeed ranking algorithm for a social media platform or a recommendation engine for a video streaming service. (Workflows for ML projects in services and consulting firms can be significantly different and will not be covered in this blog.)

Machine learning projects demand a somewhat different approach from regular software and product feature development. Building an ML feature involves experimentation with data as well as engineering heavy development. Each experiment that is successful bends future ML feature projects in new directions. But because of this, we do see a lot of friction between data scientists and engineers.

Engineers don't like having to adjust to continuously changing requirements from data scientists. Data scientists get frustrated with any hurdles that keep them from taking their improved models to production. Product teams need a feature that works well for customers and can scale at the same time.

The challenge lies in managing these projects to perfection while maintaining a delicate balance.

Harnessing friction

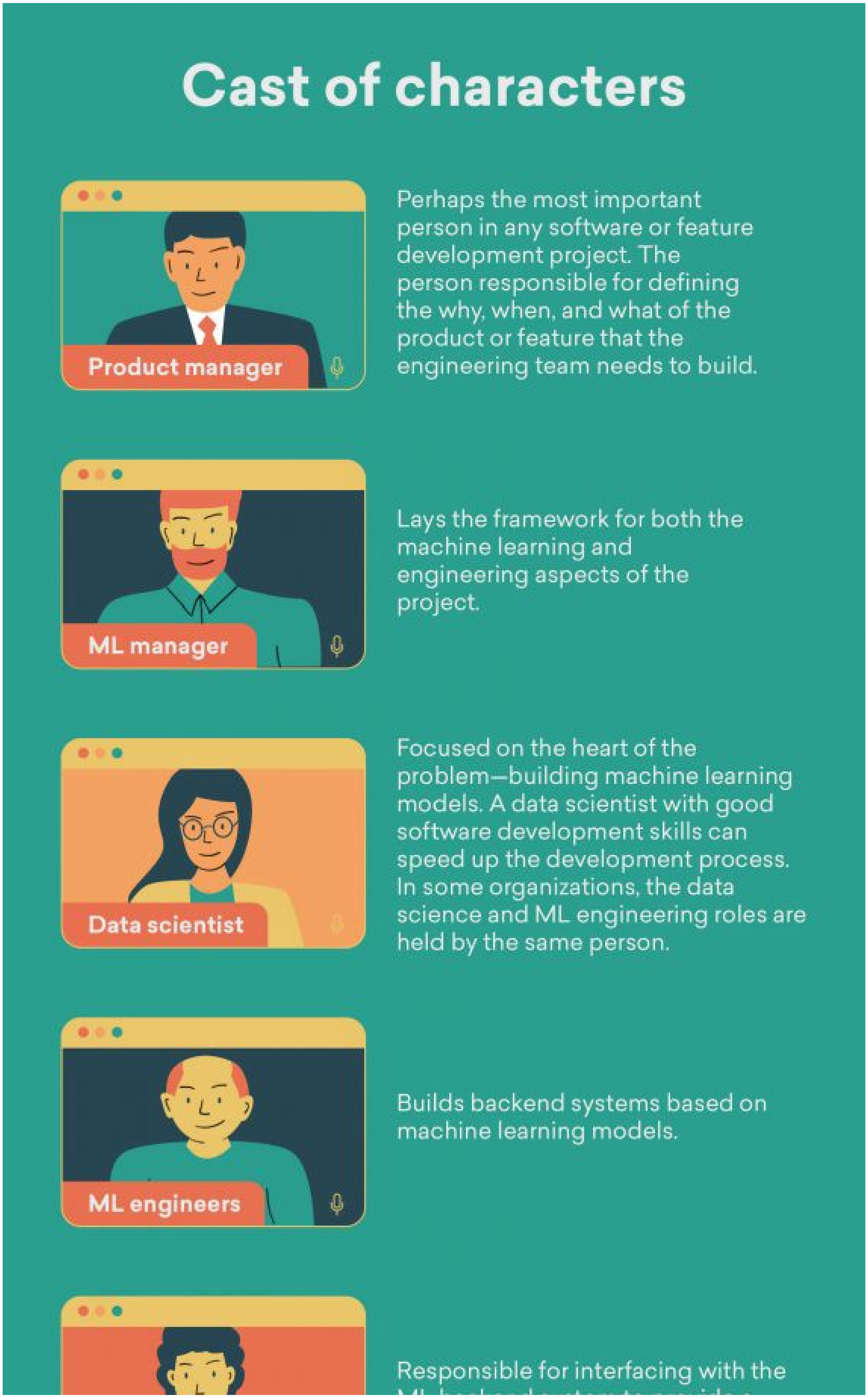
Software development projects typically involve product managers, backend developers, frontend developers and software leads, their roles broadly well-defined. Most organizations have software to track and manage their work.

Machine learning projects involve a few additional specialist stakeholders — data scientists and machine learning engineers. In some organizations, the same person straddles the two roles.

When so many highly specialized engineers come together for a project, friction is inevitable. Desired, even. Complex engagements demand friction. The challenge lies in harnessing that energy for smooth and efficient execution. You need to begin with clearly defining the roles and responsibilities for product managers, data scientists, machine learning engineers, and the sundry others who will be involved.

As the manager of a machine learning team, my role primarily entails creating a collaborative environment for ML engineers and data scientists to work with product engineers. This is to provide as much clarity and independence to each stakeholder and empower them to be on top of their game.

Many a time, a complex model built by data scientists cannot be engineered within the timelines. I need to factor in such challenges that often crop up in ML projects while defining the roles and responsibilities.



Layering project workflows

Break down any software or feature development project into bite-sized tasks that can be more easily managed and tracked. These will form the building blocks of your project workflow. Based on my experience with machine learning projects, I have crafted a workflow that would be applicable to other similar projects as well.

At Freshworks, this framework is particularly relevant to ML/AI projects such as those involving Freddy, our artificial intelligence platform. Projects that deliver predictive insights, automate repetitive tasks, and surface new opportunities so customers can harness the power of AI.

I have split the workflow into five broad categories.

1. Defining the problem

A clear statement of the task at hand is crucial for aligning all the stakeholders with the project's mission. The project manager, with help from the machine learning manager, should not just define the problem and success metrics but also translate and make them relatable to each of the stakeholders.

Task	Contributor	Owner	Description
Problem statement	Product manager	Product manager	Identify product feature based on client requirements
Problem to model	Data scientist	ML manager	Define ML problem statement so the target variable is clear

Product alignment	ML manager	Product manager	Get 'Ok' from the product team on the problem statement
User input	Product manager	Product manager	Align with the business team; understand and list pain points and usability from customers
Data source identification	Data scientist	ML manager	Identify priority data fields and carry out basic exploration; assess and get business stakeholder insights on data quality
Metric	ML manager	ML manager	Define metrics to assess model/ feature performance factoring in business usability

2. Data exploration and quality

This is the most important and yet the most ignored activity by machine learning teams. The more you dig into your data sources, the easier and better your end products will be. Data scientists and ML engineers should play tag team here.

This stage will set the rhythm for the rest of the project.

Task	Contributor	Owner	Description
Data sources	Data scientist	ML manager	Identify data sources useful to the project; begin with important features in the first iteration

Data understanding from engineering	ML engineer	ML manager	Identify data gaps along with the product engineering team; verify if fields/features are correctly published in the database/ data warehouse
Data verification	ML engineer	ML manager	Verify data counts/ quality among the different data storage systems that will be used
Cleaning scripts	Data scientist / ML engineer	ML manager	Identify/write scripts required to clean data; make data usable for building a model
Manual verification	Data scientist	ML manager	Manually verify sample data; identify basic gaps/ issues in fields of potential features/ target
Exploratory data analysis	Data scientist	ML manager	Basic data exploration (missing percentage, correlation, problem-specific plots/analysis)

3. Modeling

This stage is data science-heavy and includes all the experimentation and modeling done by the data scientist. From a product development standpoint, it is crucial to figure how and when to stop with experimentation so the engineers can take over.

The data scientist needs to ensure that the models are not too complex to be deployed within the project's timelines.

Task	Contributor	Owner	Description
Modeling approaches	Data scientist	ML manager	Identify and rate 4-5 approaches based on implementation complexity; get a benchmark with an easier approach and move to more complex ones
Model iteration	Data scientist	ML manager	Maintain experiment list and accuracy; take a broad approach that can help engineering finalize the design
Model result verification	ML manager	Product manager	Iterate the result with the product team to ensure model accuracies match product requirements

4. Productionization

This is a crucial stage where all the stakeholders meet again. Timelines, model accuracy, and engineering contracts should be revisited and finalized now. A crucial step that happens here is code handover from the data scientist to the machine learning engineers.

Once the first version of the framework/code is written, ML engineers should explain it to the data scientist. From here, any incremental changes to the model or feature engineering should be doable by the data scientist.

This is a typical point of friction between these two stakeholders.

Task	Contributor	Owner	Description

UX and frontend	Product engineer	Product manager	Develop mockups for the frontend; all stakeholders should agree with the information being conveyed to the customer
Architecture	ML engineer	ML manager	Finalize and document the architecture
Engineering contract	Product engineer / ML manager	Product manager	Define the API contract with the frontend
Devops task	Devops	ML manager	Initiate tasks around permission to services, ML ops-related work
Code transfer	Data scientist/ML engineer	ML manager	Pass on code that needs to be productionized to the ML engineer; identify complexities around scaling
Test cases identified	Data scientist	ML manager	Data scientists and ML engineers to ensure that once production code is ready the following activities are done: • Each feature vector generated in production code should match with experiment code; identify and validate each feature boundary • Accuracy of models should match between product and experiment

Production code	ML engineer	ML manager	Rewrite code according to the architecture; base framework with code is written by this step
Code update	Data scientist	ML engineer	Newer versions of the code should be written by the data scientist and reviewed by the ML manager/engineer to avoid errors

5. Quality check

Broadly, there are two aspects of quality check in this stage:

1. Ensuring model accuracies are the same between the production code and experimentations that were done earlier; and

2. Ensuring that the customer-facing interface works smoothly with the backend ML engineering framework.

Task	Contributor	Owner	Description
Code review	ML manager	ML manager	Ensure code is optimized in terms of memory usage, scalability, and adheres to architecture and product contract
Feature verification	Data scientist / ML engineer	ML manager	Verify model/ feature accuracy
			Ensure production workflow has the

Model accuracy	Data scientist	ML manager	same accuracy as one during experimentation; accuracy should not drastically improve or fall (in either case the issue needs to be debugged)
Internal POC	ML manager	Product manager	Publish internal results to clients for accuracy benchmarking
Feature QA	QA engineering / ML manager	Product manager	Final feature QA

And you're done!

I hope this framework for modeling ML project workflows is useful for teams in navigating the contours of their new workspaces.

While this blog focuses on a particular aspect of product or feature development, it offers a hassle-free template that can be adopted for other complex collaborations as well. Particularly if the only way to connect with the key stakeholders in your projects is virtual.

#freshworks #saas #machine_learning #project #management #data_science

[Co-author: Feroze Jamal]

Machine Learning Project Management SaaS Product Management Data Science