**Want to Succeed Doing ML in the Real World?**

Three tips that will make your life easier



Photo by [Robert Lukeman](https://unsplash.com/@robertlukeman?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/wallpapers/nature/waterfall?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

So, you’ve done a few ML courses and feel ready to fish for your first job in the industry, but still wondering: what are the main skills to do ML in the real world?

Working at a startup, where the pace is fast and the upside (both intellectually and financially) is high, sounds like a great idea?

Read on to learn a couple of tricks that will make your life easier and increase your chances of success doing ML in the real world.

**Context**

Machine Learning (ML) is a hard discipline. You probably know that from the online course you took, or the Kaggle competitions you participated in.

However, ML in the real world is even harder. Especially in a startup.

***Why is it so?***

Because there are more things than can go wrong doing ML in the real world, compared to an online course or a Kaggle competition.

Let me explain what are these hidden holes, and give you some advice to help avoid them.

**1. Machine Learning is the means, not the ends**

The number 1 reason for an ML project to fail is embarrassingly dumb: you do not need ML in the first place.

Machine Learning is just a (fairly complex) tool (among others) you have, to move business metrics. And more often than not, it is **NOT** the right tool.

In most startups, ML is not the core product or service, but a tool to improve the business.

In my experience, most job descriptions at startups erroneously look for “Machine Learning engineers”, when instead should be looking for “Data analysts”. A data analyst is someone comfortable analyzing medium/large datasets, extracting patterns, and quickly generating data-driven insights that can tune business processes and impact metrics. The idea is simple: in any new startup, there are tons of “low-hanging fruit” projects you can solve, using just a bit of data.

Even if your job title is “ML engineer”, do not think you need to train an ML model for every problem. You need to improve business metrics, and you probably need to use data for that. This does not mean you need to pull out a new ML model every time. You will be wasting your energy and time. Instead, focus on quick wins you can accomplish by dressing as a “data analyst”. This way you gain some time until the startup grows a bit and the real ML work enters the picture.

For example, you can analyze data offline to tune a couple of parameters in your business logic that push up metrics.

The best way to avoid all the complexities of Machine Learning is not doing Machine Learning. Sounds dumb and unappealing, but it can save you from stress, and buy you some time for another project where ML is really necessary.

**2. Where is the training data?**

If you haven’t worked on ML in the real world before, you might have never asked yourself: *where is the training data?*

And the answer is: *nowhere.*

Startups usually have a minimal data stack, comprising of 3 elements:

* **Many data sources.**These are all the production services that generate raw data. For example, imagine the startup is an e-commerce site. A data source is the google analytics service that tracks all user actions on the site. Another source would be the PayPal payment service on the site, which would send payment events.
* **The data bus** is a piece of software that channels data from all data sources into the data warehouse/lake. Startups tend to work with a paid service like Segment or Google Pub-sub, and some even adventure to run Kafka on their own.
* **The data warehouse/lake** is the place where all the data ends up, either well-organized in a database (aka data warehouse) or simply stored (aka data lake).

Once the data is in the warehouse/lake, your job as an ML engineer starts.

You often need to spend a copious amount of time writing the right queries (often in SQL) to pull the data you need to train your models.

Writing SQL queries is not the sexiest thing you’ve learned in your online courses, but it’s among the top skills you need to master to become a productive ML engineer.

Over the years, I developed a few patterns and best practices to get up to speed at this stage of the project. You can read them in my previous article, on [***how to generate training data faster and better.***](https://towardsdatascience.com/how-to-generate-training-data-faster-and-better-e84d05884dc0)

**[How to generate training data: Faster and better](https://towardsdatascience.com/how-to-generate-training-data-faster-and-better-e84d05884dc0" \t "_blank)**

[Best practices to boost your productivity as a Machine Learning engineer](https://towardsdatascience.com/how-to-generate-training-data-faster-and-better-e84d05884dc0" \t "_blank)

[towardsdatascience.com](https://towardsdatascience.com/how-to-generate-training-data-faster-and-better-e84d05884dc0" \t "_blank)

**3. How do I show *others* that my model works?**

Ok, my model looks good, now what?

In a business environment like a startup, a Machine Learning model has 0 value as long as it stays confined inside your development environment.

Unless you deploy the model and expose it to the rest of the tech stack, for example, through a microservice and a REST API, you will have no impact on business metrics.

However, to get there, you first need to prove to the rest of the team that:

1. Your model actually works.
2. Your model is better than the status quo.

For example, imagine you develop a binary classifier that is able to predict if a customer will churn in the next few days. The marketing team can use such a model to target potential churners and keep them engaged.

When you are in the development phase, you use standard evaluation metrics for classification problems, e.g.

* precision-recall
* f1-scores
* accuracies,
* and confusion matrices.

With these metrics, you get a sense if your model has any predictive power at all and whether it is worth deploying and using in production. However, these metrics are not the end of the story, because:

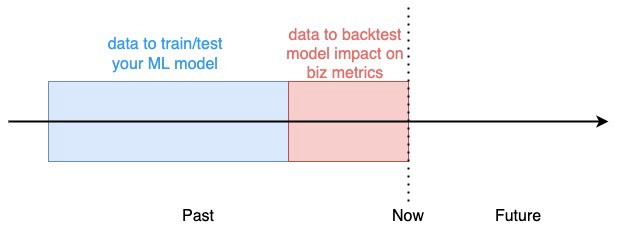
1. They are **hard to understand by non-technical stakeholders**, e.g. the marketing team that will use the model.
2. They are **not easily interpretable from a business perspective**, e.g. *by how much are we going to increase retention if we use this churn prediction model?*

Because of this, if you want to have a real impact on a startup, you will need to walk 2 extra steps, which are:

1. Backtesting your model, and/or
2. A/B testing your model.

**Backtesting your model**

To backtest a model, you use historical data up to a given date *X* to train a model, and then estimate what would have happened, in terms of business metrics, if the model had been used from date *X*onwards.



Backtesting your model impact (Image by the author)

For example, in our churn prediction system, we could use historical data up to 2 weeks ago and then estimate how many more clients we would retain in the last 2 weeks if we had used the model.

Model backtesting is a standard strategy used in the financial industry, to assess trading strategy performance and risk management. It is fast to implement, and inexpensive because you already have all the data you need.

However, it is hard or even impossible to implement in situations where feedback loops exist between the ML model and the environment where the model operates. In our churn prediction model, for example, it is impossible to know how would have different clients reacted if they had been flagged as churners by our model and targeted by the marketing team.

In these cases, you need to take one step further and run an A/B test on your model.

**A/B testing your model**

To A/B test our churn prediction system you would **randomly** split the users into 2 groups:

* **A group** (aka **control group)**. This segment of users would be treated as usual by the marketing team, which means, no special re-targeting efforts triggered by the churn prediction model.
* **B group** (aka **test group)**. These users would be flagged as churners/no-churners by our model and targetted by the marketing team accordingly.



A/B testing your ML model (Image by the author)

The split is random, to guarantee there are no significant behavioral differences between the 2 groups, which would bias the results of the test.

The test runs for *N* days and at the end, we compare the overall churn rate between the 2 groups.

If you observe a significant reduction in churn rate in the B group compared to the A group, you have solid evidence your model works and can be deployed to the entire user base.

Otherwise, you need to either improve the model or move to another problem.

And although it slows you down, A/B testing is still the most effective methodology to convince your colleagues your model should be deployed.

So, stick in there and you WILL see the results.



Jagoda and Kai (Image by the author)

**Wrapping it up**

So, the next time you find yourself in front of a challenging real-world problem, ask yourself:

1. Do I need ML at all?
2. Where is my data?
3. And how will I show that my solution works?

And believe me, it WILL make a difference.