**5 Reasons Your ML Model Isn’t Making It to Production**

Accuracy metric not being understood? Try creating a lift plot



Photo by [Max Duzij](https://unsplash.com/es/@max_duz?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

Ok, so you’ve spent months building your machine learning model. You’ve split the train/test sets, ensured there isn’t any data leakage, and tuned the hyperparameters to finally achieve a model with 99% accuracy! Now all you have to do is run it in production and it will improve the customer experience.

You take the final product and show it to the product managers, emphasizing how well the model is performing and that all that’s left to do is to put it into place. Months go by, then a year, and your model still hasn’t been prioritized in the roadmap. What gives?

This isn’t an uncommon occurrence for data scientist work and it can be very demotivating. Below we’ll discuss 5 reasons why many models don’t make it to production even in companies that really want to make use of your models, and what we can do to ensure that most if not all of our models make it to production.

**1. You left it as a notebook**

Jupyter notebooks are amazing for most data science work. They’re well suited to exploratory analysis, prototyping, quick development, and rapid scripting. However, they’re not generally great for writing production-level code. The ability to run cells out of order and having access to all variables across the notebook can lead to code that goes against best practices. It is often better to pythonise and modularise your code into scripts with multiple functions, which also works with traditional unit testing and version control more smoothly.

Note that this is not always the issue, as Netflix uses notebooks directly in production <https://netflixtechblog.com/notebook-innovation-591ee3221233?gi=a7ada81eb58c> with the assistance of tools like [nbdev](https://nbdev.fast.ai/).

**2. No one is sure how to actually productionise it**

A common issue is that a notebook that runs the model on a dataset you ingested is complete, but you need the model to run inside the product in real time. However, the main codebase also isn’t in Python so you can’t import pandas or perform the operations you use to clean and process the data. The software engineers on the team aren’t versed in Python and the DevOps engineers have their hands full with production issues. Sound familiar?

In this case, you may need to branch out into MLOps to make sure the model actually gets to production (we can deploy our model as an API using flask and docker as I’ll cover in a later article). At smaller companies with less mature data science functions, this will often be the path of least resistance. The alternative is to make a business case to purchase an MLOps tool or to ensure strong product and business stakeholder buy-in as we’ll see below.

**3. No one else is invested in it**

The best and most direct way to ensure your models make it to production is to get product managers involved and invested in them. The point of machine learning models is to solve problems, so by developing models to solve a key issue for product managers, we can quickly and easily get stakeholder investment whilst knowing that our solution has a tangible impact. This can help to make sure that model deployment is added to the roadmap and that you get valuable cross-team engineering team time.

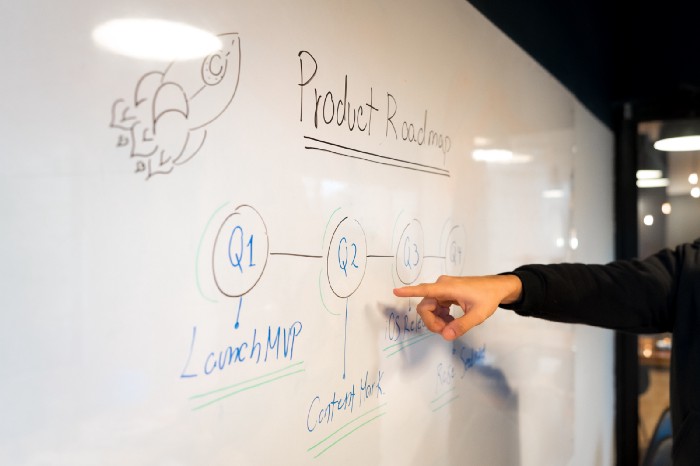


Photo by [Slidebean](https://unsplash.com/@slidebean?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

The caveat to this is that you’ll need to work on stakeholder management during the development of the model. It can also be particularly difficult to make time estimates for data science projects, which will be needed in order to be included as a part of the roadmap. It may also be difficult to implement monitoring or CI/CD in this way.

**4. No one else has any idea what it’s for**

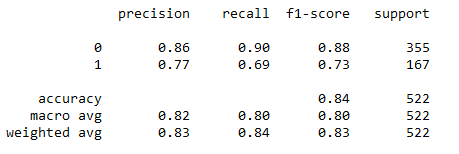
Throughout this article, the key idea in general is to ensure that the model being built will be used by the end stakeholder. You’ll need to work with them to understand how they’ll use the result of your work, inside the product itself or otherwise. There’s no point in creating an API that predicts the risk of a customer applying for credit if your credit risk team has no rules in place to utilize that output.

In the past, I’ve seen some amazing work that solved a key problem and even deployed end-to-end left unused because the deployment wasn’t inside the tool that the operational end-users actually used. The solution actually just ended up gathering dust. This could have been avoided with just a little discussion with the actual users themselves — don’t let this happen to your work!

**5. No one else has any idea what its output means (create a lift plot)**

This a classic discussion that I’m sure you’ve heard many times before, but even if our model has 99% accuracy this actually means very little to other data scientists, and even less to business stakeholders. Here let’s go back to our credit risk case where we can predict the probability of default by splitting the classes into ‘default’ and ‘no default’ to create a classification problem. Then we can utilize the model’s predict\_proba method to get the probability of default. Credit risk is a common imbalanced dataset problem as the class with no defaults will heavily outweigh those with defaults (unless the credit risk analysts are doing an awful job!). In such cases, our model could potentially reach 99% accuracy by only predicting non-defaults for the whole population, which is performing great metric-wise but is actually a pretty bad model!

A better metric to use for classification might be a classification report, which gives a confusion matrix of the precision and recall for the positive and negative classes. This would flag the performance of a bad model which only predicts no defaults as for defaults (1) it would have 0.0 precision and recall due to no true positives.



Example [sklearn.metrics.classification\_report](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html). Image by author

To better communicate the meaning of this to business stakeholders we need to translate this into a business context. Most machine learning models have positives (rejecting correct defaulters) and negatives (incorrectly rejecting a good applicant), so what would be the actual impact on the bottom line? The most important metrics here are usually revenue and profit so let’s have a look at this for our credit risk example.

First, we create a lift plot by bucketing the probability for each data point and checking if the actual classification behavior reflects that of the bucket. Below we show an idealized case where the bucket default proportions are very close to the probability. The bucket sizes can be adjusted depending on the size of the dataset.

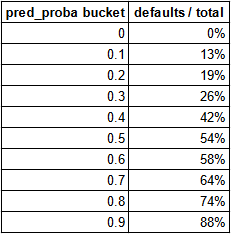
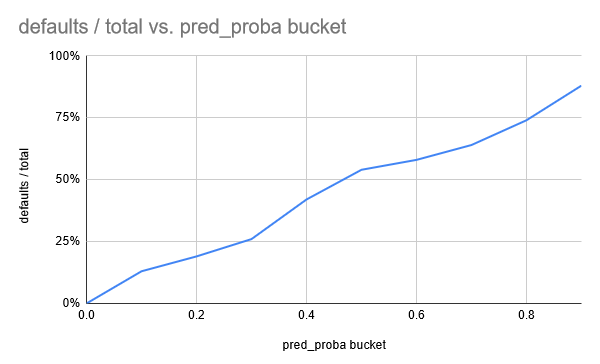


Table for a very good classification model. Image by author



Plot for a very good classification model. Image by author

Now that we have the lift plot we can communicate the impact of using our model to set default probability thresholds. For example from the table above, if we reject credit applications above the threshold at 0.9 probability, then 88% of our rejections will be correct, and we will lose revenue from our 12% of incorrect rejections. This can now be easily transformed into monetary values and turned into a discussion with the business on what to do going forwards depending on if revenue or profitability is more important.

**Final Thoughts**

Many data scientists do amazing work to solve difficult problems, but can have difficulty getting their solutions to production. We discussed that a common result of this is the need to pick up MLOps-type skills, but that the most important thing is collaborating with product managers and end-user stakeholders on what you’re building and how it will actually be used. To better enable you to decide how a model will be used, transforming common metrics such as accuracy or AUC into actual business impact is the way to go.