# ML System is not an ML Model

*The core of this article is based of my reading for the Designing Machine Learning Systems book, plus some other resources!*

***"In 2020, two ML Engineers at Google, looked at 96 cases where I large ML pipeline at Google broke. They reviewed data from over the previous 15 years to determine the causes and found out that 60 out of theses 96 failures happened due to causes not directly related to ML. Most of the issues are related to the distributed systems, workflow schedule, data pipeline, ... etc."*** *Page 288. Chapter 8: Data Distribution Shifts and Monitoring.*

Welcome to my second article, first is Clean/Good Code/Architecture. Why?. During Hackathon April 2023, I'm looking into Machine learning space by reading *Designing Machine Learning Systems* book. Why? Well, most of my experience is in building general software systems with focusing on the backend side and did not work with ML before, and since I joined Yelp and AMD team in February 2022 I started hearing more about ML, not a surprise as my team is responsible for the Market Dynamics in the Ads space. So I started developing interests in this area, and given I'm a backend engineer, I looked into it with engineering perspective, not applied/data scientist one.

In this article I will summarize my takeaways from the "Designing Machine Learning Systems" book, and why ML system is not only ML Model, and what are the other components/steps we should consider while building ML system. As mentioned this is just my understanding, and definitely not a guid to someone who building an ML system, but hopefully it'd be a good intro for non-ML people to understand that they can contribute to ML systems even if they are not ML experts (applied/data scientist or ML Engineer), and also might be helpful for people start working on the ML models to understand why other parts are important as well.

**In practice we can divide building ML systems in 4 steps, after scoping:**

## Data Engineering

In this step we need to be aware of data types(Historical data in storage, and streaming data in real-time transport, e.g. Data Pipeline at Yelp). Each data type may need different processing, Batch Processing vs. Stream Processing.
You may do not need to look into data sources much, but if something is on fire, it will be critical to look into the inputs for your system, and data is the most critical one!

The data referred to in previous paragraph is the raw data, but we need to do feature engineering to extract the features and important information to be used for the Model training. As mentioned in the Practical Lessons from Predicting Clicks on Ads at Facebook paper, **"Having the right features is the most important thing in their developing ML models".** Oh, I have too many features, which one to use, good news, there are some ways to test the feature importance.

There are some common feature engineering operations like 1) Handling missing values, 2) Scaling (backend engineer? it is not the scaling you know 😃), feature scaling! 3) Discretization, ... etc and there are different ways for handling each of these operations.
Generally adding more features leads to better performance, but sometimes not for free, more features you have 1) more cleaning up effort, 2) can cause overfitting, 3) increase the used memory, 4) increase inference latency, and 5) more infrastructure work to make those features available.

## ML Model Development and Evaluation

This is the core/heart of the ML system, developing the model itself which requires domain-specific knowledge and skills, different skillset from backend /general engineer. I have to admit that chapters talked about this stage in the book where hardest chapters for me to understand which is make sense I'm not an ML engineer nor applied/data scientist!

Anyway I'm going to skip this stage, as the goal of the article is to discuss the non-ML model steps.

## Deployment

This maybe the most part that backend engineers can relate. After crafting the ML model, we need to think about how to make it and the data available in prod and be connected to the other systems. In this stage, where we build the data pipelines, setup some batch processing or stream(real-time) processing, design API calls to other services, add logs. Oh, think about latency! should we add cache, should we challenge our applied/data scientists friends to tune some parameters to make the model much faster!

With discussing with our applied/data scientists friends, we will think about batch prediction vs. online prediction. Both of them have pros and cons, e.g. latency, model performance, .. etc.

## Monitoring and Continual Learning

Finally, we deployed the model, ok, will forget about it and if we do not do anything, model performance remains the same. It is a myth! The author of the book dedicated 2 chapters to talk about the monitoring and continual learning.

Why do we need monitoring? A) ML specific failures, and B) Software System failures.

Will not talk much about the general Software System failures, those are the common ones, e.g. Dependency failures, Deployment stuck, Crashing, Latency, ....etc.

The ML-specific failures are more interesting! 😃 Let's talk a look on some of them:

**Production data different from training data:** Our Model has trained using specific training data, so what could happen if the production data is different from training data? Our backend OE dashboards will not catch it.

**Edge cases:** monitoring edge cases importance vary based on what your model does, recommender system is not like self-driving car!

**Degenerate feedback loops:** this can happen when the predictions themselves influence the feedback, which in turn, influences the next iteration of the model. a.k.a next model output is the future input, this would cause popularity/exposure bias.

**Data Distribution Shifts:** Most probably, we will retrain the model! 😃

After looking into some ML-specific failures, it is clear that we need to have specific monitoring additional to the the generic monitoring for any system. We need to monitor 1) Model accuracy-related metric. 2) Predictions. 3) Features, and 4) Raw data/Inputs. It is important to have logs, dashboards and great to have alerts as well for those metrics.

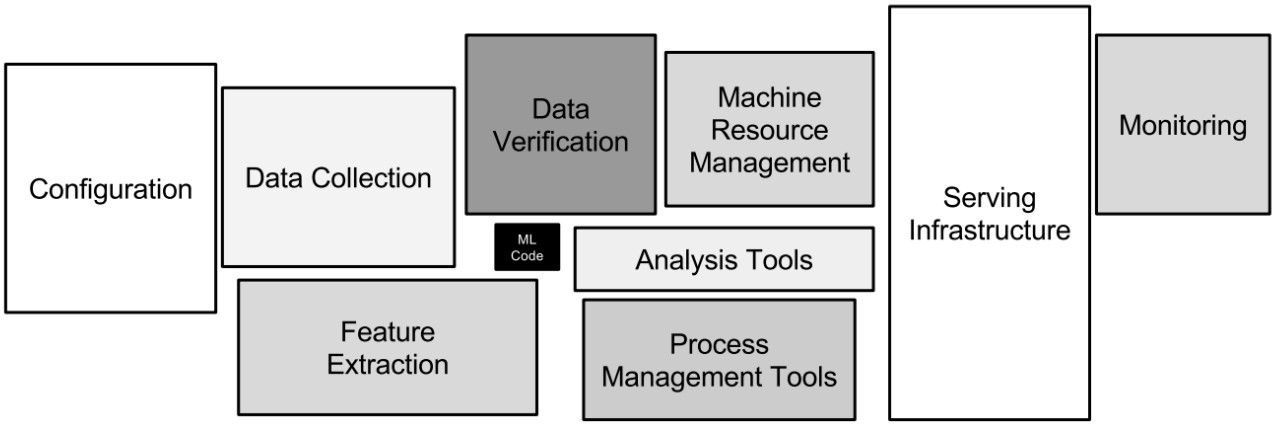
For the **Continual Learning** part. Model updates could be 1) Model iteration, add new features, or change the model architecture. 2) Data iteration, need fresh data! The book talked about different strategies for retraining , we have to have reason to retrain our model, our change it retaining frequency, it is not beneficial to retrain the model daily instead of weekly, and we see no performance gain, we just throwing money on not needed infra!

It is worth to mention that some models that need to retrained very often, e.g. every hour, may face some challenges, 1) Fresh data access (Yaay, cool backend problem to solve 😃), 2) Model Evaluation challenge, to make sure this update is good to go.

Related to the second challenge, there are some ways to deploy/test in prod, e.g. 1) Shadow Deployment. 2) A/B Testing. 3) Canary Release. 4) Interleaving Experiments.

## Okay, it is not only ML Model

It is clear now, that many other steps and work included in the building ML system that not in particular ML model/code/algorithms.




Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex. *From Hidden Technical Debt in Machine Learning Systems*

For me as Backend Engineer, I think I can easily understand and can do most of the work for the stages for building an ML System, expect the core step, the ML Model Development and Evaluation and might part of extracting the features, here is where our applied/data scientist friends do their magic 💡. Building big ML Systems would require collaboration between backend engineers and applied/data scientist, or may ML Engineer who can do everything 😉.

## Sounds there are some common things!

As you may feel now that many of these steps will be applied to any ML model, the chapter 10 ML Ops. is talking about that, and why building ML Platform is a make sense decision for companies to take. The ML Platform generally would help with Model deployment, Model store(definition, parameters, dependencies, data, ...), Feature Store. I hope you head of Yelp ML Platform before.

# Resources:


- *Designing Machine Learning Systems book*
- ML System Maintenance: Hidden Technical Debt (This is a great wiki showing some real examples at Yelp from postmortems, CEPs, JIRA issues)