Data Science in production

Lecture 1: Introduction

Alaa BAKHTI

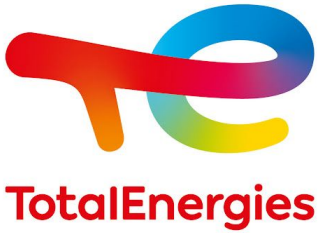
Who am I?

EPITA 2018 : did the piscine 🙀🥴🤕🙈

Teacher and advisor @ EPITA
- Advisor for AIS & DSA
- Teaches the courses:
  - Recommender Systems
  - Time Series
  - Data Science in production

Machine Learning Engineer @ OCTO Technology

Interested in Data Science, Software craftsmanship => AI in production



What about you?

- Your profile
- Any knowledge or experience with ML powered applications?
- Your expectations for the course

The course

- 8 sessions
- Grading:
  - Project - 60%
  - 2 practical works - 30%
  - Participation - 10%

Syllabus

1. Code versioning with Git and environment management in Python
2. Introduction
3. Coding best practices 1
4. Coding best practices 2
5. Versioning in a data science project
6. Quality validation for model building and integration
7. Model serving and deployment strategies
8. Model monitoring and retraining
9. Machine learning delivery best practices (bonus)

Some rules

- No showing up late to the courses !
- No computer during theory part
- No cheating in the practical work & project: 2 students last year were caught
  => disciplinary board

Introduction

ML in research vs. in production

	Research	Production
Objectives	Most accurate model	Generate value to the user
Computational priority	Fast training	Fast inference
Data	Static	Constantly shifting
Fairness	Good to have	Important
Interpretability	Good to have	Important

Source : Stanford MLSys Seminar Episode 5: Chip Huyen

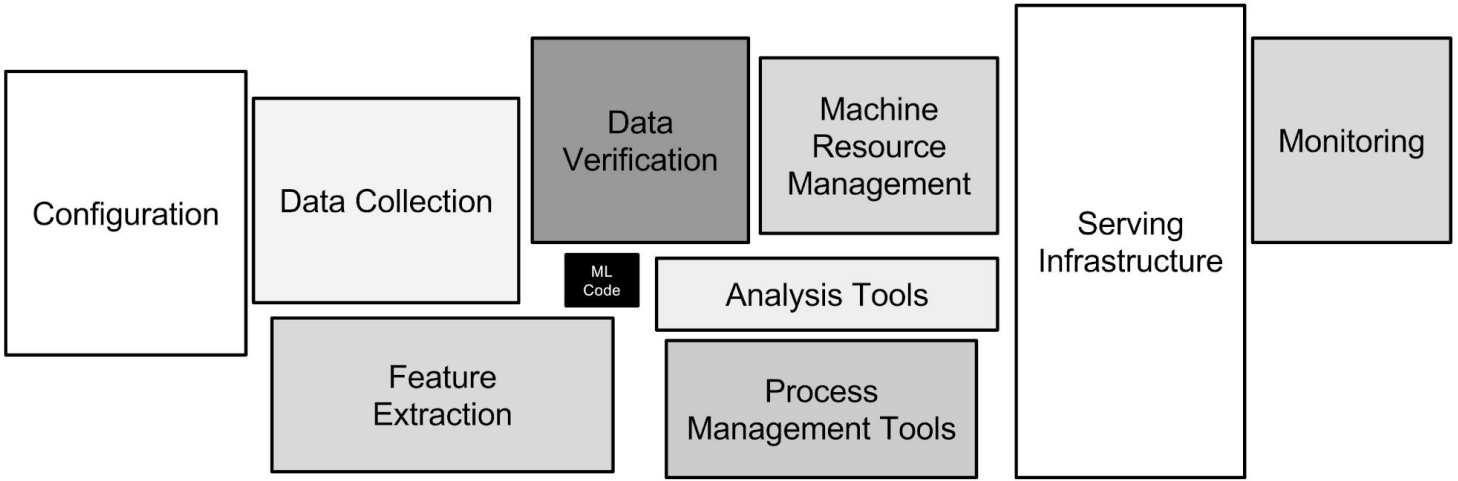


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Hidden Technical Debt in Machine Learning Systems (Google, NIPS 2015)

ML-based software vs. traditional software

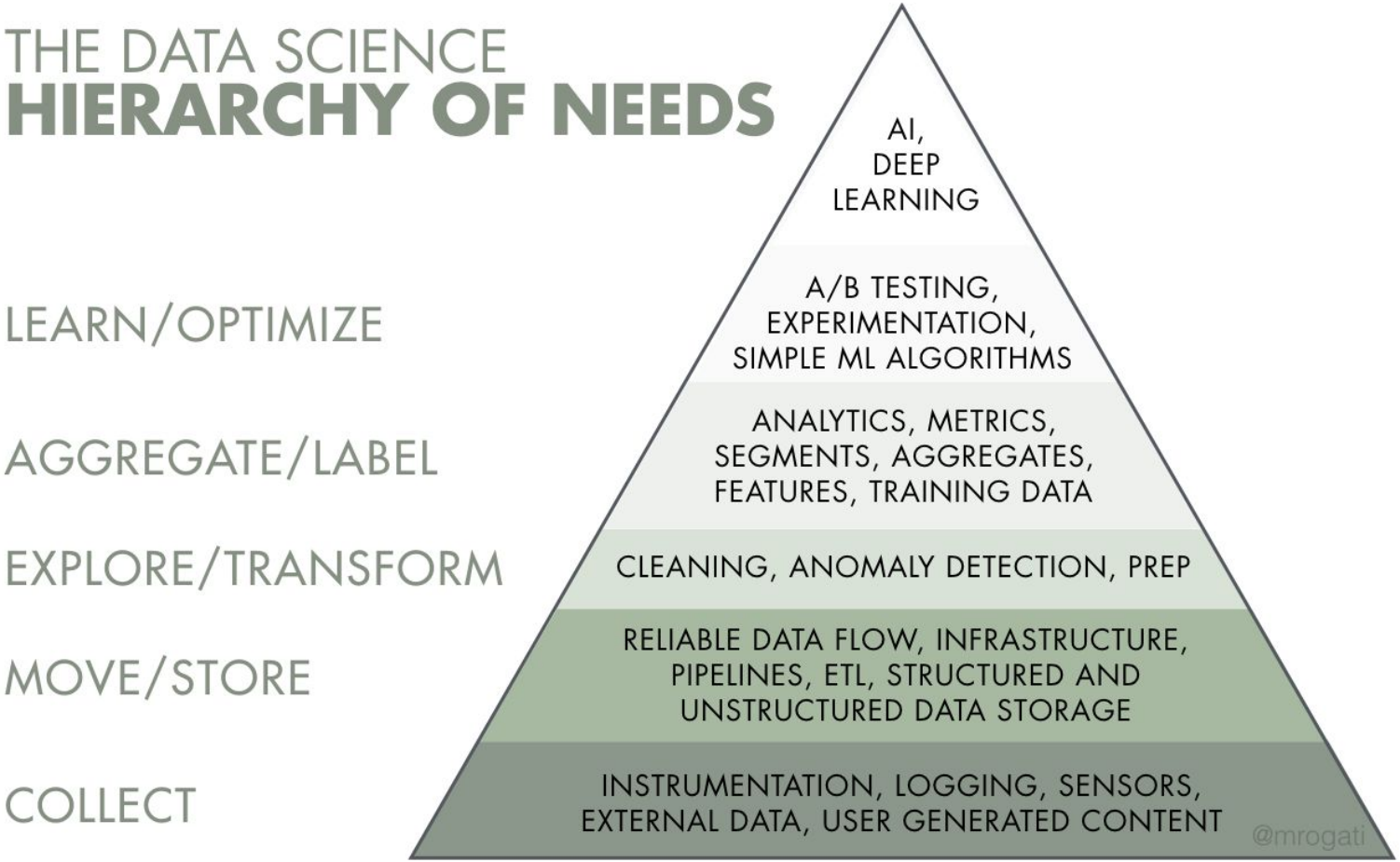
More than half century of maturity in the Software engineering field.
Why we don’t treat ML based software as a traditional software ?

- In ML based software, we have 3 artifacts to manage : Code, Data & ML model
  - **Data** : may shift over time (product deployment in a new region, users changing behavior, etc)
  - Manage and improve the data to improve the model (*test, monitor, version, etc)*
  - **Model** : will degrade over time and will need to be adapted (updated, replaced) (*test, version, debug, monitor, etc)*
- Uncertainty & estimations
  - Hard to estimate the required time (1 month? 2 months?) to develop a model that meets the business expectations (e.g. 90% of recall) >> How to define the product road map ?!

“The process for developing, deploying, and continuously improving Machine Learning applications is more complex compared to more traditional software, such as a web service or a mobile application. They are subject to change in three axis: the code itself, the model, and the data. Their behaviour is often complex and hard to predict, and they are harder to test, harder to explain, and harder to improve.”

- martinfowler.com, Continuous Delivery for Machine Learning

Machine learning powered applications



The AI Hierarchy of Needs (Monica Rogati, 2017)

Different types of Data Science project

1. Integrate a ML functionality in an application
   - The application contains different features not related to ML.
   - The ML feature represent a small part of the final application
   - The product value is delivered even if the ML feature delivery fails.

2. Deliver an ML powered application
   - The main features of the application are based on ML.
   - Risk on the delivered value because of the uncertainty of the ML features (big work to do on the scoping phase)

Different roles in a Data Science project

Team composition

- Product Owner (PO) and / or Subject Matter expert (SME)
- User Experience designer (UX)
- Data engineer
- Data Scientist / Machine Learning Engineer
- Software engineer
- IT operations (Ops)
- etc

\+ Users

\+ Project sponsors

End-to-End Machine Learning workflow

Definition of business needs and use case framing

Data preparation

- Integrate data from different sources and transform it into curated datasets for model building

Model building

- Prepare dataset, extract features, train and evaluate the model (offline evaluation)

Model validation

- Validate the model raw performance, meta performance (inference time, CPU, RAM) and bias (ethical considerations).

End-to-End Machine Learning workflow

Model serving & integration

- Definition of the model serving strategy and integration in the application

Model deployment

- Deploy the model in production so that the users can access it
- Evaluate it using metier and business KPIs (online evaluation)

Model & data monitoring

- Monitor the model in production to detect performance decay

Model retraining

- Upon environmental change, retrain the model on an up to date dataset

Some ressources

- Continuous Delivery for Machine Learning (CD4ML)
- Hidden Technical Debt in Machine Learning Systems, NIPS 2015
- visenger/awesome-mlops: A curated list of references for MLOps
- Made With ML

Project

Team: 3 or 4 people per team

Subject: You are free to choose the use case you want (energy consumption prediction, etc)

Requirements

For your project to be validated, you need to cover all the mandatory requirements along with 1 of the 2 choice requirements

Mandatory requirements

- Model as a service serving strategy (FastAPI, Flask, etc)
- User interface (streamlit, Flask, etc): the user should be able to:
  - Make on-demand predictions
  - View past predictions (made by the prediction job (see below) & queried from the database)
- Prediction job (Airflow): make scheduled predictions each n mins and save them in a pgsql database

Project

- ~~Experiment tracking with MLflow tracking during mode training: use the same pgsql database for predictions and the MLflow backend store) (for more information)~~

Choice requirements: you need to choose either *data ingestion and validation pipeline & monitoring* or the *re-training pipeline*

Data ingestion and validation pipeline & monitoring

- Ingest data each minute from a source to another (from a folder to another folder for example)
- Validate the quality of the ingested data (great_expections) & implement the alerting system
- Prediction monitoring dashboard (Grafana)
    - Distribution of the predicted data features VS the distribution of the production model training data features
    - Distribution of the predictions (minutely, hourly, dalily, etc)
    - etc

Project

Re-training pipeline

- Detect when there is a drift in the production data (the production model training data distribution and the ingested data distribution are no longer the same (the Kolmogorov–Smirnov test) (**Bonus**)
- Train a new model each n minute or when a data drift is detected
- Use the MLflow model registry for model governance
- Deploy the new model when a new model transitions to the MLflow registry *Production* stage

Milestones

- Group member formation and group name choice: in 1 week
- Use case definition: V1 in 2 weeks, V2 in 3 weeks
- Follow-up sessions: each 2, 3 weeks after the use case definition
- Project defense: during the last session (project presentation + demo)

Project

Note

- For the project, you need to
  - Use git (with branching) to collaborate together on the project: I'll check the git commits and branches during the follow-up sessions
  - Environment management with miniconda or any other tool (pipenv, etc)
- During the follow-up sessions, each group member will present the work he/she has done and the work that he/she'll be doing in the next weeks
- ~~The final project grade will be composed of a group grade (60%) and an individual grade (40%)~~
  - the group grade depends on the final results and on how the group worked together during the project period
  - The individual grade depends on the contribution of each group member to the project
  - The 2 grades will be based on the notes I'll be taking during the follow-up sessions

Project - Ressources

Made With ML

Evidently AI - Open-Source Machine Learning Monitoring

- Real-time ML monitoring: building live dashboards with Evidently and Grafana