i4talent

MLOps

using FastAPI and MLFlow

i4talent DATA SCIENCE

ML, more than just models

"When designing an ML system, people who haven't deployed an ML system often make the ***mistake of focusing too much on the model development*** *part and* ***not enough on the model deployment and maintenance part***"

Let’s create an awesome ML system!

At the end of the day you will be able to:

- Understand what **ML** is in **production vs research**
- What **MLOps** is and why it is so important
- Think and design **ML systems in a holistic approach**
- **Deploy a ML model** with a **REST API**
- **Versioning** of ML models and log the results
- Understand **model and data drift**
- **Monitor** and evaluate ML **models**
- Build **trigger(s)** for **continual (stateful) learning**

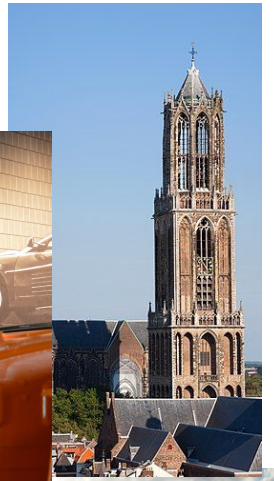
But first, introductions!

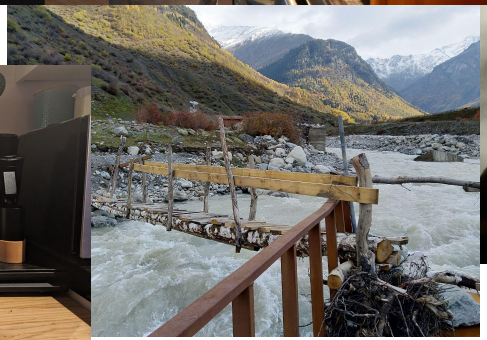
Coffee geek

Imported Utrechter

> 90k minutes of spotify a year

Travel without a plan





i4talent DATA SCIENCE

But first, introductions!

- Grew up in Zoetermeer with two younger sisters
- Graduated from TU Delft (fell in love with programming and data during Civil engineering Master degree)
- Machine learning enthusiast
- Father of a girl (2.5) and a baby boy (1)
- Formula 1 lover (since 1998)
- Loves reading and coding in his spare time (when the kids are sleeping…)
- Can’t work(out) without music

i4talent DATA SCIENCE

Agenda

- Introduction to ML as a system
- Model serving
- The model of the day
- MLOps
- Model monitoring and data drift

ML as a system

ML difference in research and production

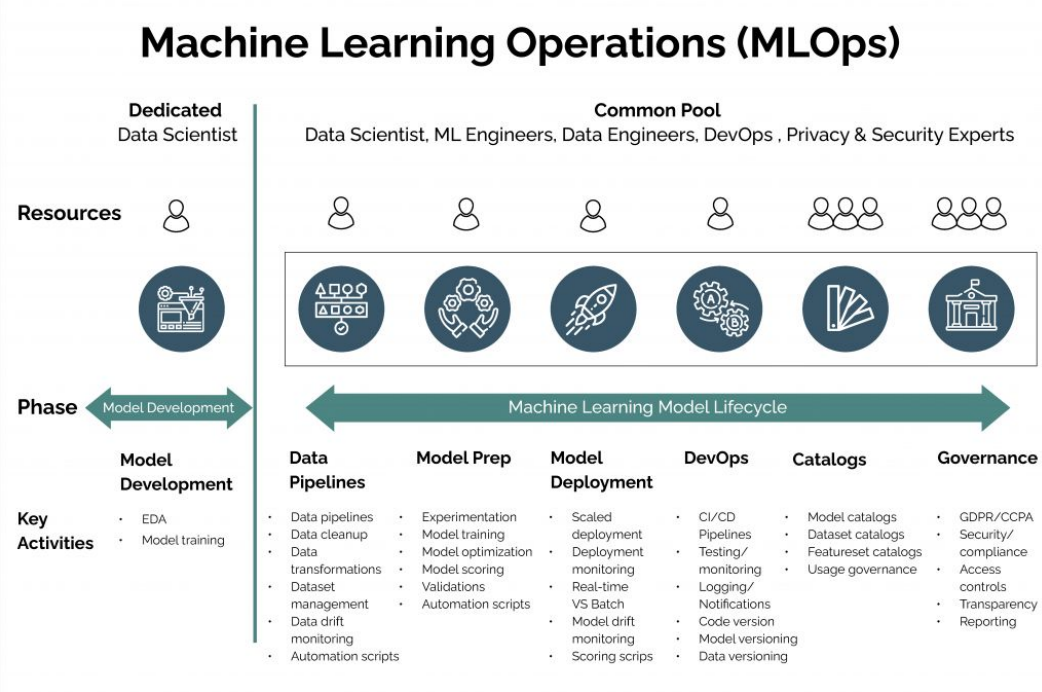
	Research	Production
Requirements	State of the art model performance on benchmark datasets	Different stakeholders have different requirements
Computational priority	Fast training, high throughput (how many samples you can process)	Fast inference, low latency (how long it takes for each sample to be processed)
Data	Static	Constantly shifting
Fairness	Often not a focus	Must be considered
Interpretability	Often not a focus	Must be considered

Which algorithm to choose in ML systems?

When considering what model to use for your problem, it's important to consider not only the model's performance like in research by metrics such as accuracy, F1, log loss, et cetera but also:

- **How much data is needed?**
- **How long does it take to (re)train?**
- **What is the inference latency?**
- **Interpretability of the model**

Data science vs MLOps



i4talent DATA SCIENCE

But it all starts with...

Framing business problems as ML problems

Warming up exercise: framing ML problems

Suppose we are working at Netflix and business development has asked us to use ML to improve revenue.

- **How can we frame this business objective/ problem in a task that ML can solve?**
- **Can we come up with ML objectives that align with the business objectives?**

Business objectives → ML objectives → ML Models

Four requirements for ML systems

1. **Reliability**
   The system should continue to perform even in the face of adversity (e.g. software faults)

2. **Scalability**
   The system is able to support growth in ML models and traffic

3. **Maintainability**
   The system should have versioning for code, data and artifacts

4. **Adaptability**
   To adapt to shifting data distributions and business requirements, the system should have capacity for discovering performance improvements and allowing updates without service interruption

Machine learning is much more than ‘just’ a model… and deployment…

We need a holistic systematical approach.

What else do we need besides model development?

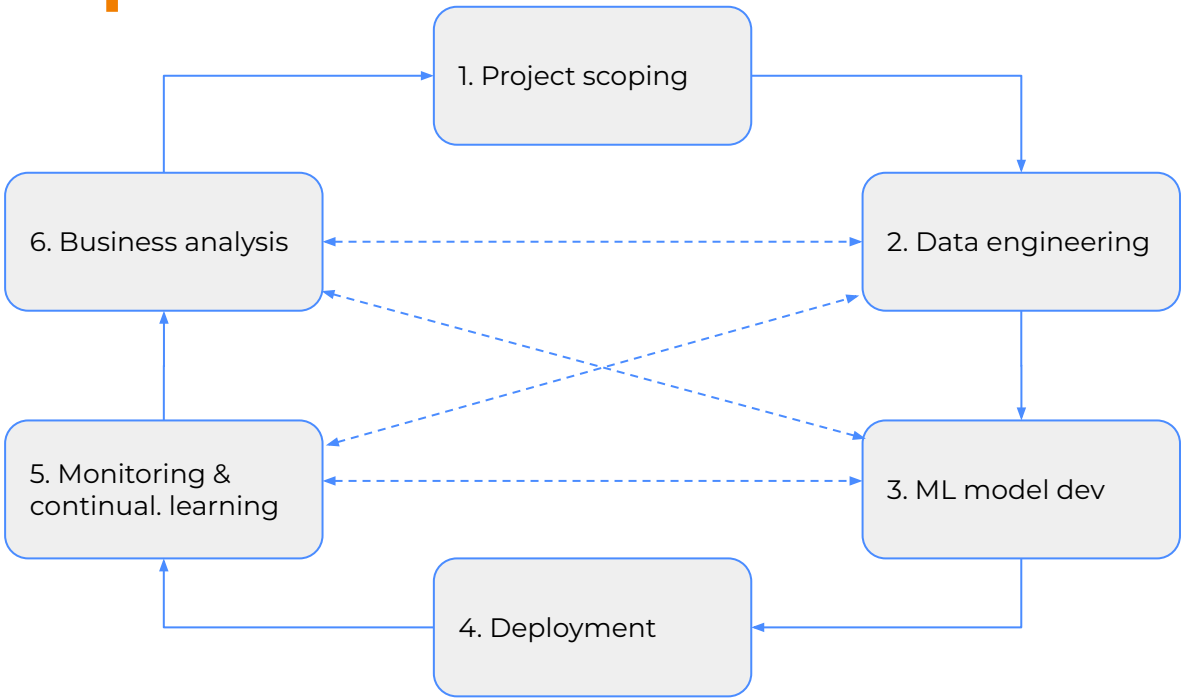
Machine learning is much more than ‘just’ a model (and deployment)…

We need a holistic systematical approach.

What else do we need besides model development?

1. Aligning ML objectives with business objectives
2. Data engineering
3. Feature engineering
4. Model training
5. Model evaluation (offline)
6. Model deployment
7. CI/CD
8. Model monitoring and data drift detection
9. Triggering for retraining

Never-ending process of ML system development



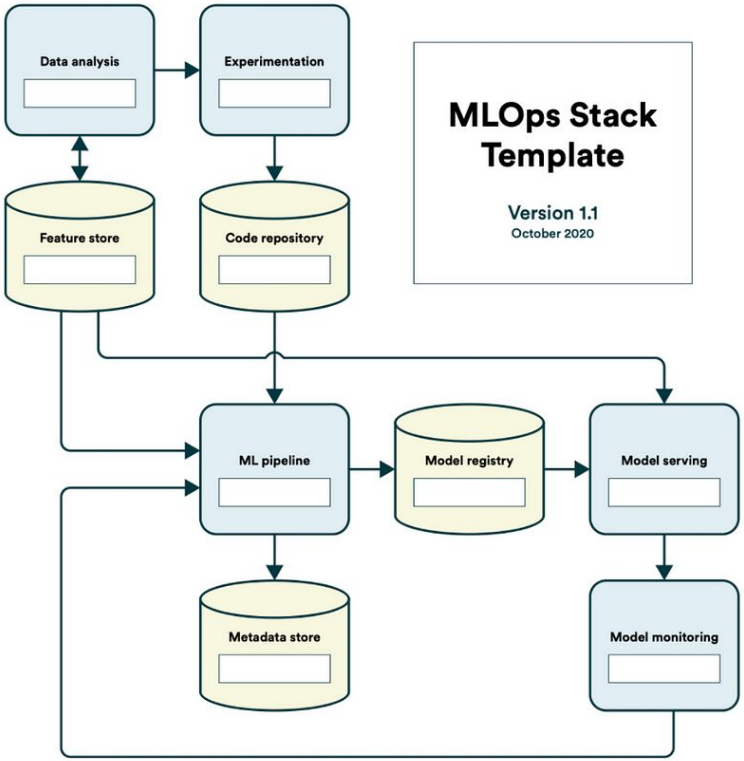
Hands-on part 0!

Assignments: https://bit.ly/3S9OhRf

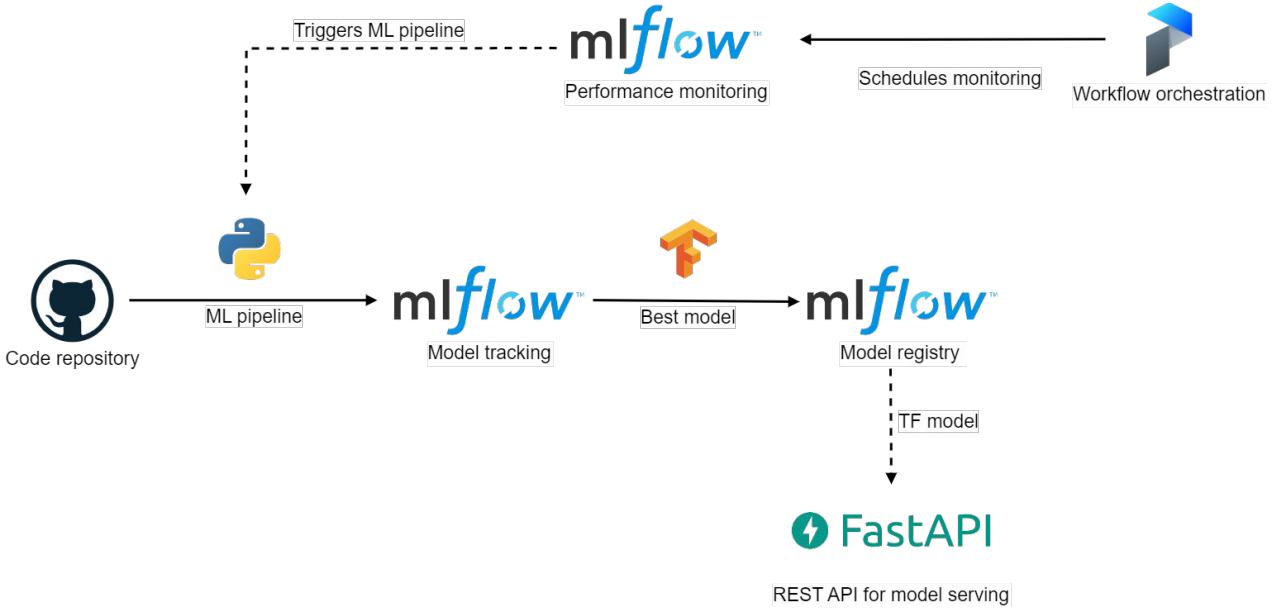
Slides: https://bit.ly/40jZL6Q

Work on part 0 of the assignments

MLOps technology architecture template

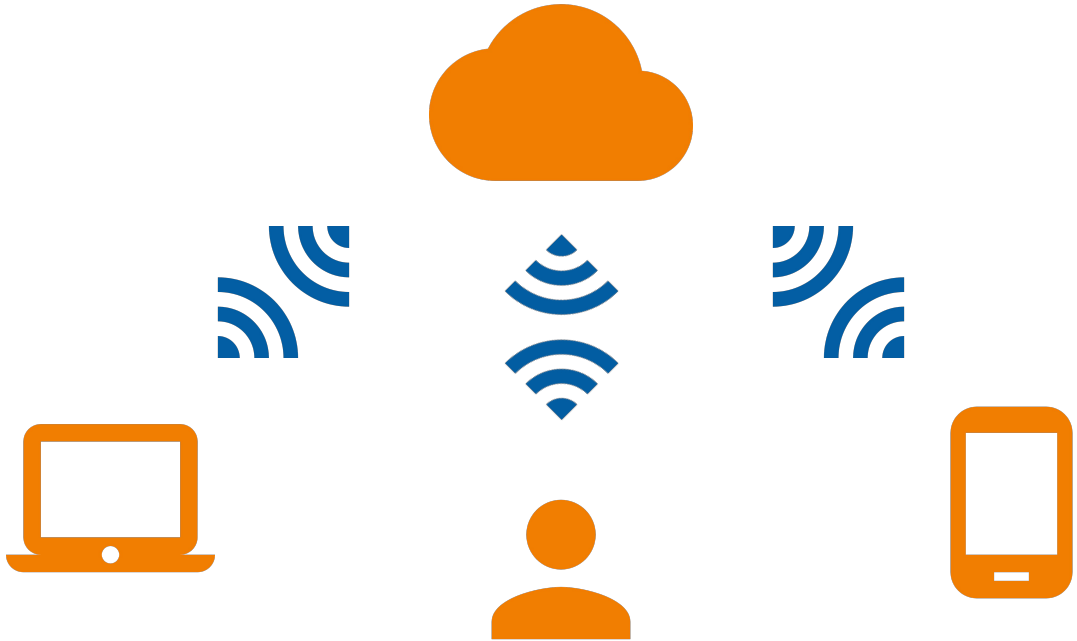


MLOps architecture for today



Model Serving

A model is useless if it cannot be interacted with.



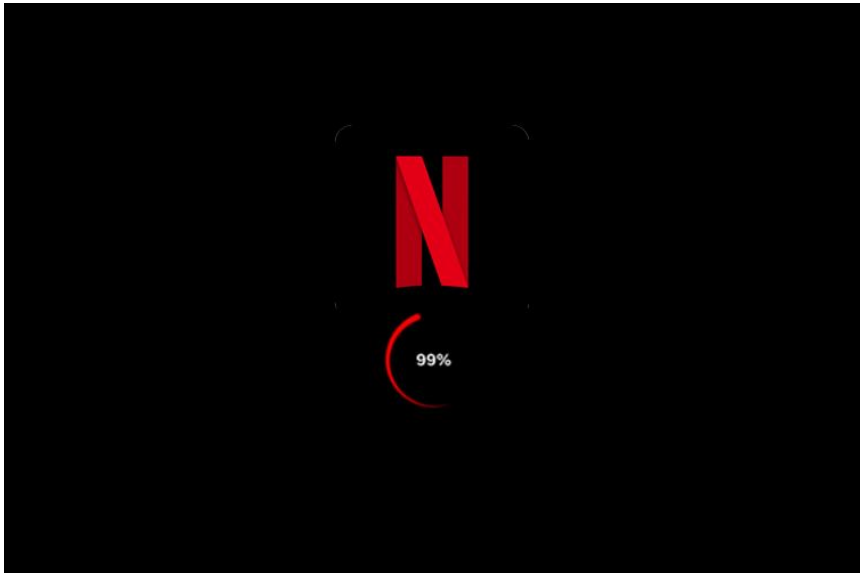
What kind of model deployment flavours are there?

Model deployment can be considered as the process of exposing a trained ML model in a production environment to be used by the rest of the world for making inferences.

- **Local offline predictions** (in a Notebook)
- **Batch predictions** (predictions on a file of multiple instances)
- **Online (real-time) predictions** (on-demand predictions through HTTP calls)

So we work at Netflix... what kind of model deployment do we need?

So what do we need to use our recommender system in real-time?



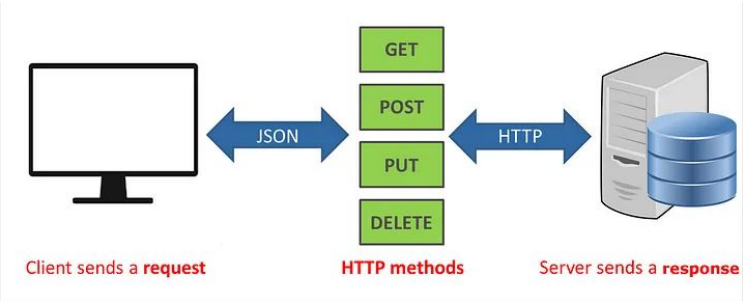
A (fast) API!

API = Application Programming Interface

Used to simplify communication with an application

Most times we are talking about a Web API:

http://127.0.0.1:8000/some/path



At the end of the day you will have build one yourself.

The model of the day: retrieval model for recommendations

Recommender Systems

“We are leaving the age of information and entering the age of recommendation.” - Chris Anderson 2004 (“The Long Tail”)

Recommender systems make predictions based on (user) historical data; they predict item preference (e.g. movies) based on past behavior (e.g. ratings).

What are recommender systems?

Types of data (feedback) to build a recommender system for:

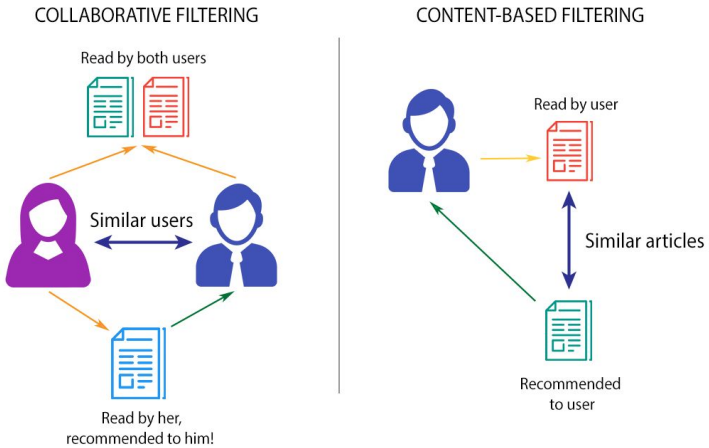
1. **Explicit feedback**
   e.g. ratings or likes

2. **Implicit feedback**
   e.g. amount of clicks, streams or percentage completed

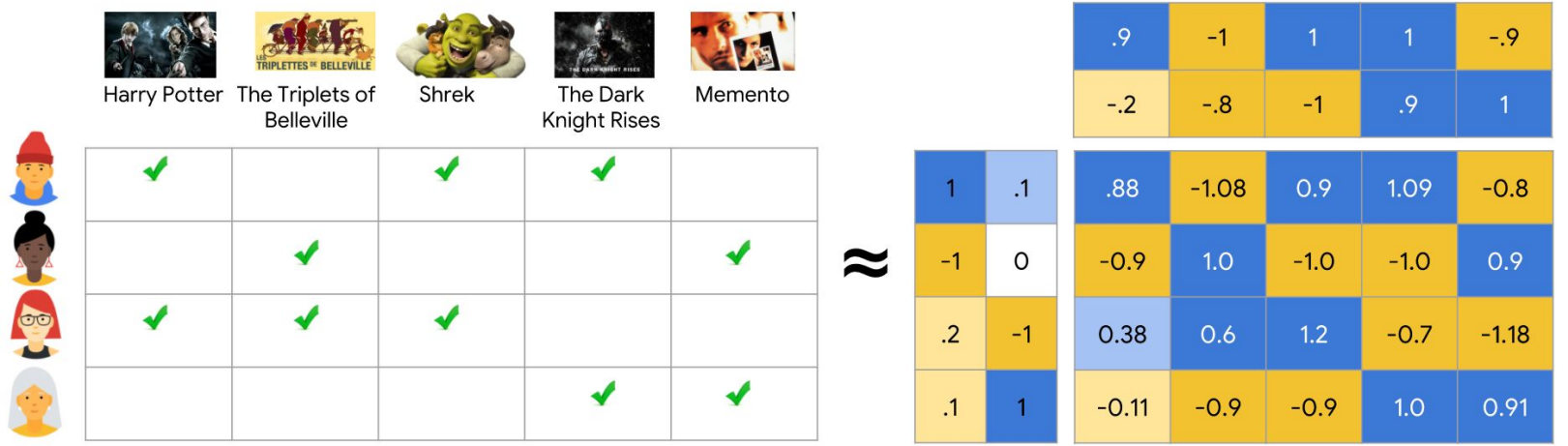
i4talent DATA SCIENCE

Types of recommender systems

1. **Content-based**
   Based on item data (e.g. description or features). The model uses for example cosine similarity to look for similar items.
2. **Collaborative filtering**
   Uses similarities between users and items simultaneously to provide recommendations (e.g. matrix factorization)



Example of a collaborative filtering recommender system



✓		✓	✓	
	✓			✓
✓	✓	✓		
			✓	✓

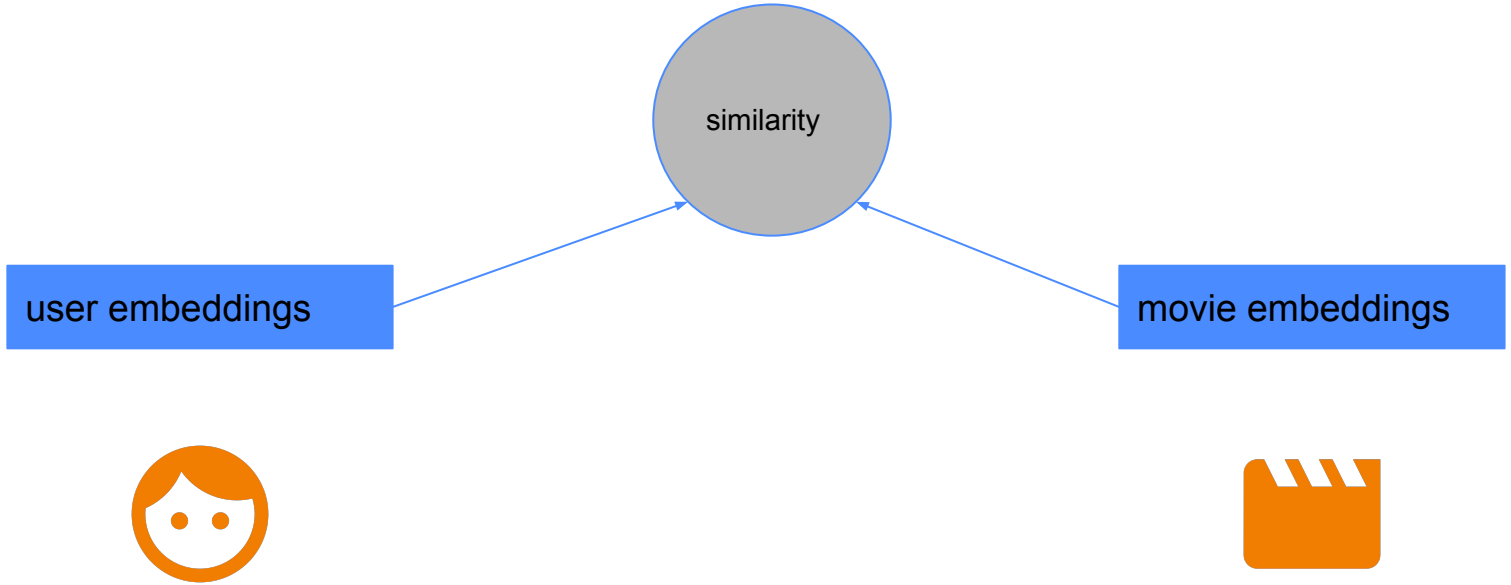
1	.1
-1	0
.2	-1
.1	1

.9	-1	1	1	-.9
-.2	-.8	-1	.9	1

.88	-1.08	0.9	1.09	-0.8
-0.9	1.0	-1.0	-1.0	0.9
0.38	0.6	1.2	-0.7	-1.18
-0.11	-0.9	-0.9	1.0	0.91

i4talent DATA SCIENCE

Collaborative filtering using a two tower model



For a more advanced system you can add a second model

We focus on the retrieval stage of a recommender system

Good for quickly filtering irrelevant recommendations, but not very good at estimating actual ratings

A second deep neural network is often used to fine-tune the result of the retrieval stage

Hands-on part 1!

Repository: https://github.com/i4talent/fastapi_harvest

Assignments: https://bit.ly/3S9OhRf

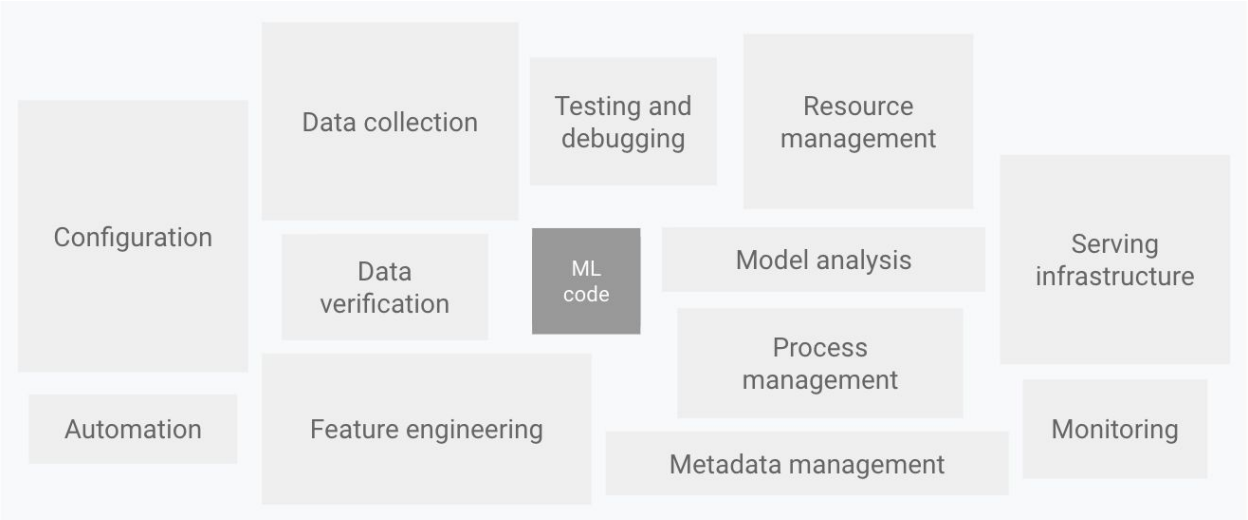
Slides: https://bit.ly/40jZL6O

Work on part 1 of the assignments

Switch to the branch **solutions_homework** for a codebase with completed homework and some hints.

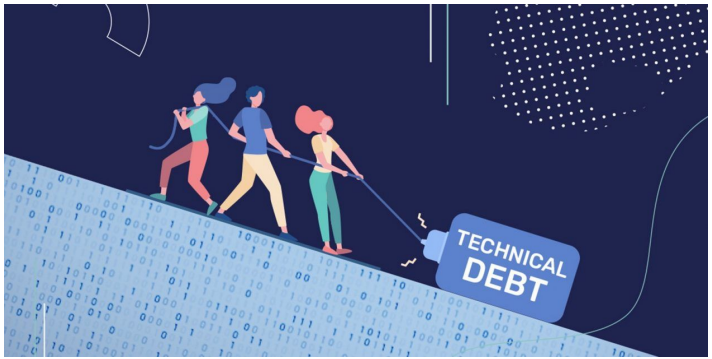
i4talent DATA SCIENCE

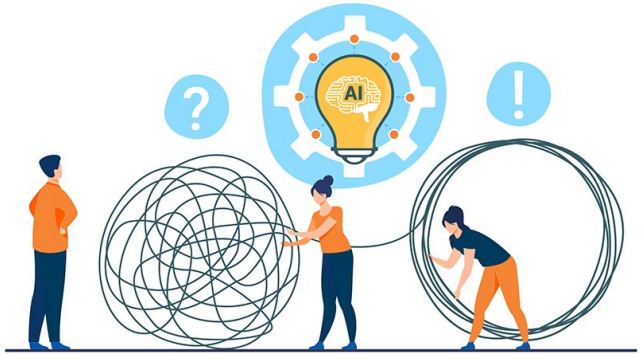
ML systems consist of many sub-systems



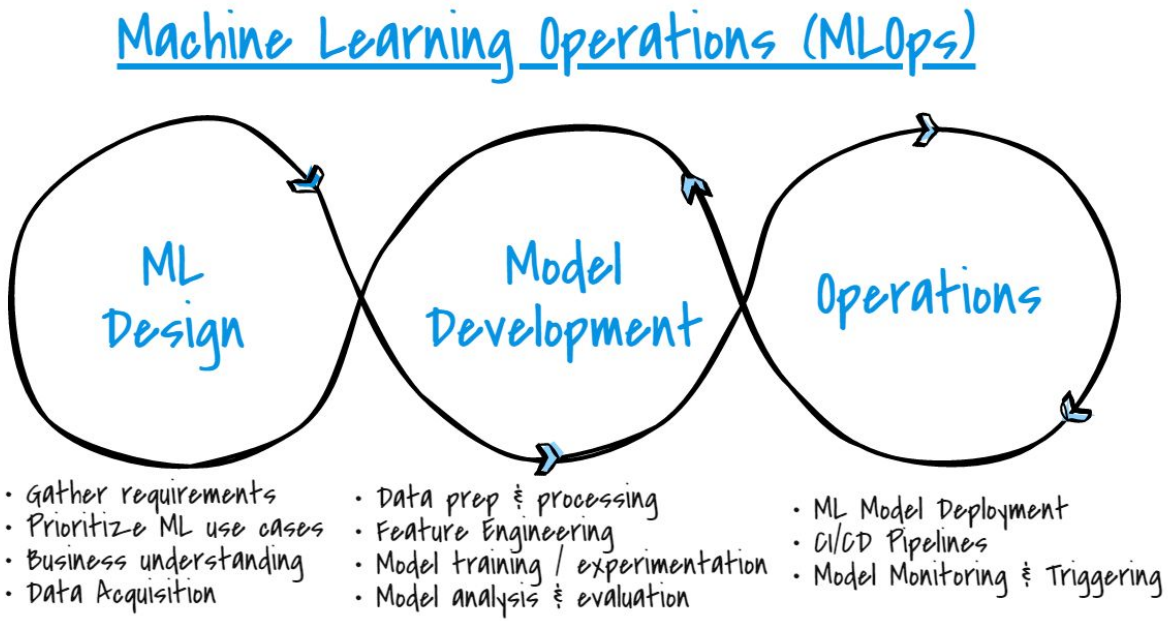
And therefore is prone to technical debt

Technical debt is the ongoing cost of expedient decisions made during code implementation, which tend to compound.





Therefore we need a continuous (and automated) loop for code, data and model changes



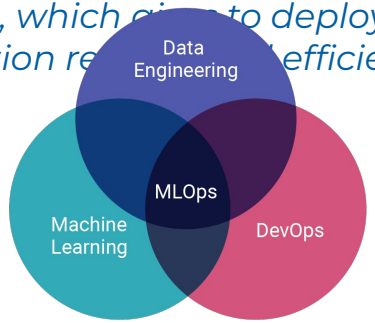
MLOps (Machine Learning Operations)!

So we need MLOps in order to efficiently manage end-to-end lifecycle for machine learning models **to ensure reduced time and cost of pushing models into production** and therefore strive to **avoid "technical debt"** in ML systems.

MLOps != ML + DevOps

MLOps can be defined as:

MLOps is a set of practices that lies in the intersection between ML, DevOps & Data Engineering, which aim to deploy and maintain ML systems in production reliably and efficiently.



i4talent DATA SCIENCE

MLOps (Machine Learning Operations)!

So MLOps is **more than just ML + DevOps**.

MLOps also tackles ML-specific challenges like:

- Data & Artifact versioning
- Model (re)training & evaluation of models
- Production monitoring to ensure the performance of the model with new/ unseen data
- Dynamic scaling of computing power (infrastructure) in production

MLOps principles

The main objective of MLOps is to **avoid the "technical debt"** involved in developing and deploying ML systems, to ensure that machine learning is:

- **Reproducible**
  It ensures that given the same input, each phase of data processing, ML model training, and ML model deployment should yield similar outputs.

- **Collaborative**
  The collaboration is usually between data scientists, ML engineers, business analysts and IT operations professionals. MLOps encourages teams to make transparent the whole process of creating an ML model, from data extraction through model deployment and monitoring.

MLOps principles

- **Scalable**
  MLOps enables organizations to scale in order to address critical issues by making ML initiatives more efficient and effective.

- **Continuous**
  1. Continuous Integration (CI): testing code and containerizing it
  2. Continuous Delivery (CD): automatically deploy model prediction services
  3. Continuous Training (CT): automatically retraining ML models for re-deployment
  4. Continuous Monitoring (CM): monitoring production data and model performance

Hands-on part 2!

Repository: https://github.com/i4talent/fastapi_harvest

Assignments: https://bit.ly/3S9OhRf

Slides: https://bit.ly/3IUPAxX

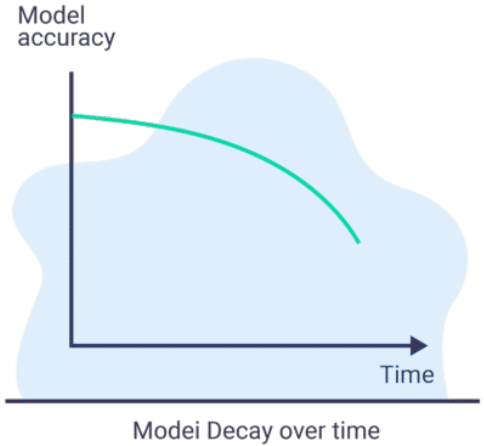
Work on part 2 of the assignments.

Switch to the branch **solutions_part1** for the solutions to part 1 and some hints and placeholders for completing part 2.

We got a perfect model! But it performs poorly in production after some time...

A common problem in ML is model decay.

Model decay is the gradual decline in the performance of a ML model over time.



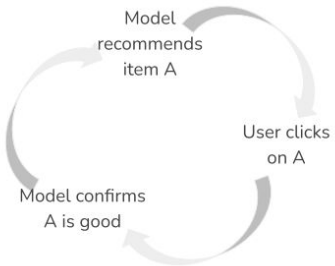
Modei Decay over time

Reasons why ML system may fail

1. **Software system failures**
   - Dependency failure
   - Deployment failure
   - Hardware failures
   - Downtime

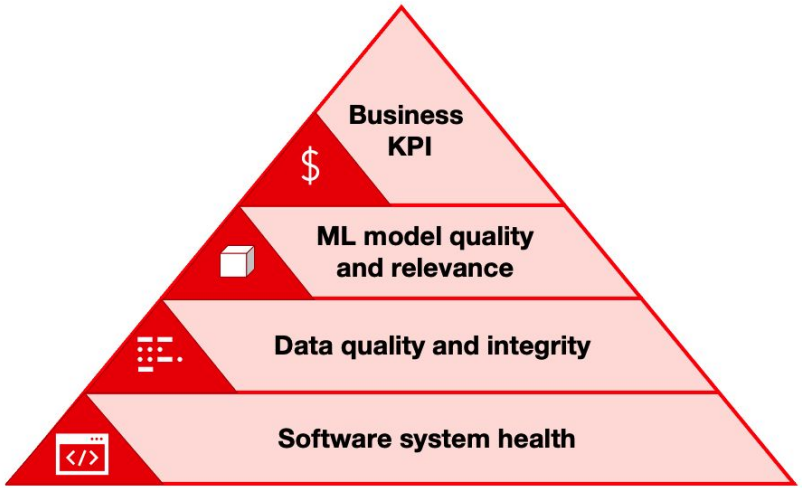
2. **ML-specific failures**
   - Data drift
   - Edge cases (data samples that cause the model to make catastrophic mistakes)
   - Degenerate feedback loops

- Originally, A is ranked marginally higher than B -> model recommends A
- After a while, A is ranked much higher than B



Over time, recommendations become more homogenous

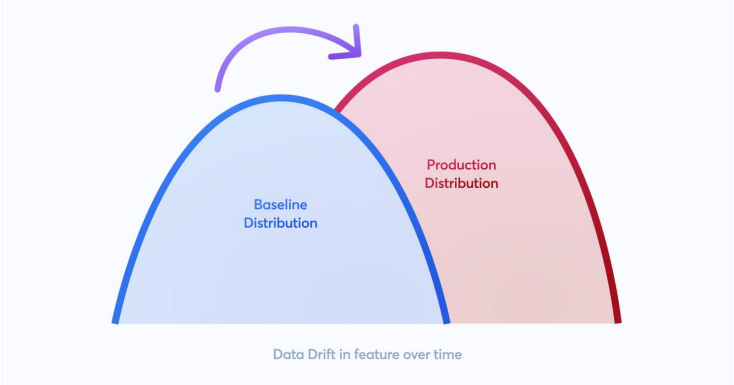
Hence we need monitoring to detect ML-specific failures



One of the most important causes of failures: data drift

In machine learning, model drift refers to a change in the underlying distribution of the data that a model has been trained on, leading to a decrease in its performance on new, unseen data.

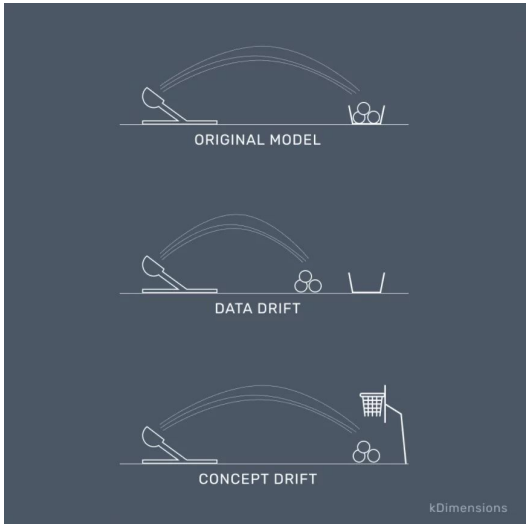
For example, a model trained on pre-covid data may not perform as well on data during the Covid19 pandemic due to changes in the underlying distribution of the data.



i4talent DATA SCIENCE

Types of drift

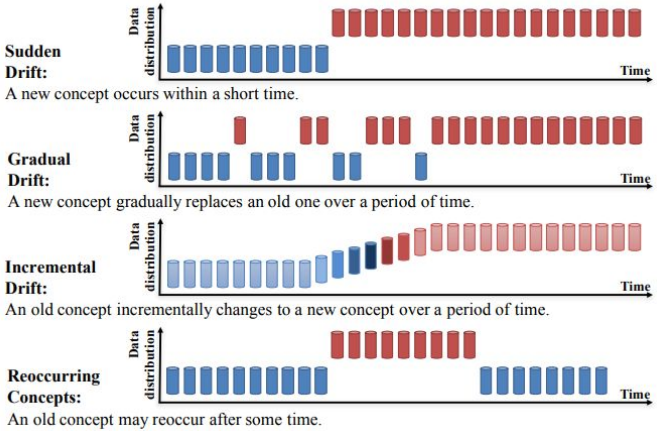
1. **Concept drift(posterior drift)** → P(Y|X) changes but P(X) remains the same
2. **Covariate shift (data drift)** → P(X) changes but P(Y|X) remains the same
3. **Label shift (prior probability shift)** → P(Y) changes but P(X|Y) remains the same



1. Concept drift (posterior drift)

Concept drift is the change in relationship between the independent and target variable. This occurs when the underlying concept or task that the model is trying to learn changes over time. For example, a model trained to detect fraudulent credit card transactions may experience concept drift if the type of fraud changes over time.

Concept drift can be divided into four categories:



2. Covariate shift (data drift)

Covariate shift is the shift in independent variable(s).

This occurs when the distribution of the input variables changes over time, but the underlying concept or task remains the same. For example, a model trained on data from one geographic location may experience covariate shift if it is deployed in a different location with different distribution of input variables.

3. Prior probability (label) shift

Prior probability shift is the shift in the target variable.

For example, a model trained on a dataset where the classes are balanced may experience prior probability shift if it is deployed on a dataset where one class is much more prevalent than the other.

Techniques to detect data drift

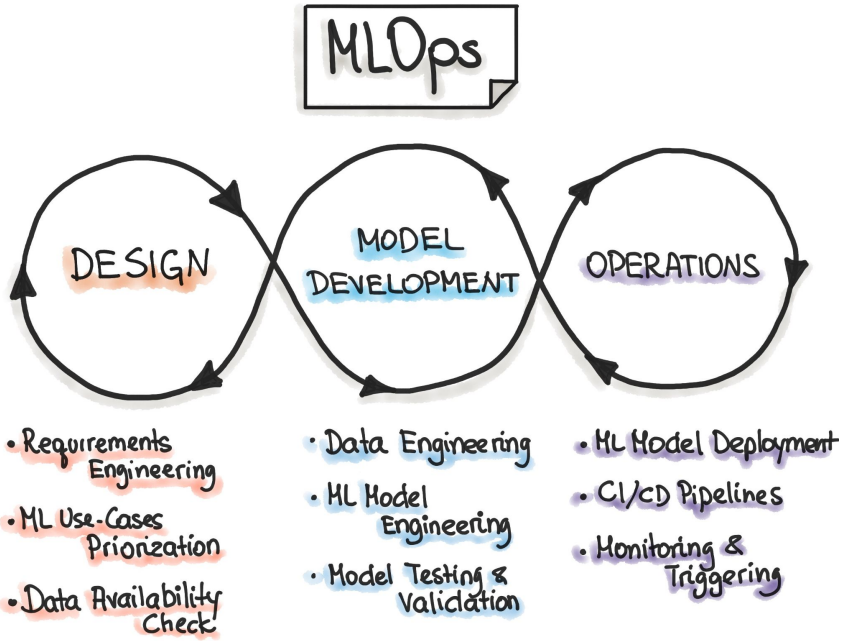
There are different methods to detect model decay and data drift.

1. **Monitor model performance**
   Detecting deviations between test and production performance (e.g. MSE, RMSE, accuracy, precision, recall)
   → might be not the best solution where there is a large time gap between time of prediction and obtaining feedback (e.g. marketing campaigns)

2. **Descriptive statistics and changes to distributions to detect covariate shift**
   - Kolmogorov-Smirnov test
   - Population Stability Index (PSI)
   - Wasserstein distance
   - ...

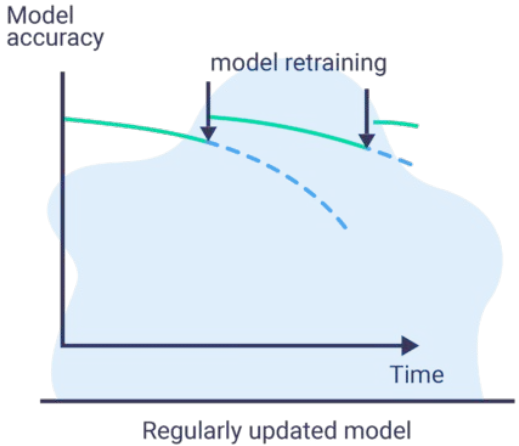
The missing piece in closing the MLOps loop...

Continual learning!



Therefore we need to retrain our model regularly

However, retraining our model on a fixed time interval may not be the best option.



Four stage of Continual learning

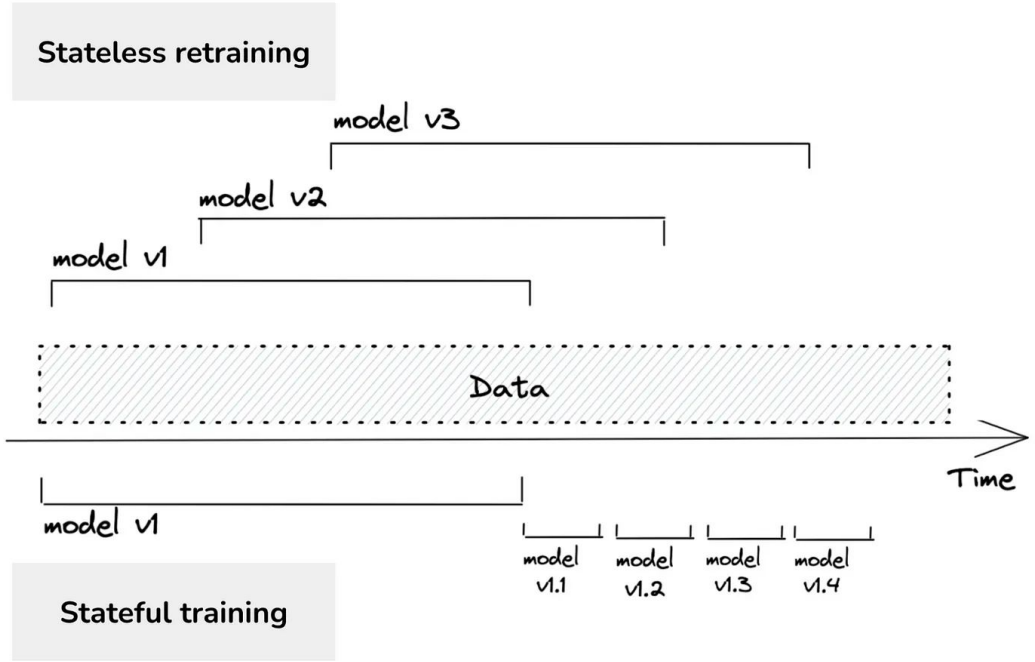
Stage 1: Manual, stateless retraining

Stage 2: Automated retraining

Stage 3: Automated, stateful training

Stage 4: Continual learning

Stateless vs stateful learning



Continual learning

Allowing stateful training - the model continues training on new data and the current model is fine-tuned.

→ It allows to update your model with less data (e.g. only new data from last week)

Hands-on part 3!

Repository: https://github.com/i4talent/fastapi_harvest

Assignments: https://bit.ly/3S9OhRf

Slides: https://bit.ly/40jZL6O

Work on part 3 of the assignments.

Switch to the branch **solutions_part2** for the solutions to part 2 and some hints and placeholders for completing part 3.

i4talent DATA SCIENCE

How could we improve this?

Run in a Docker container

Include a ranking model for better predictions

Add automatic retraining

Add endpoints to monitor model performance

Add SSL/TLS using nginx

Deploy it to a cloud platform

Or something else?

Thanks!

Questions?

bas.dekan@i4talent-ai.nl

https://www.linkedin.com/in/bas-de-kan/

niels.hoogeveen@i4talent-ai.nl

https://www.linkedin.com/in/nhoogeveen/

i4talent DATA SCIENCE

Feedback formulier

https://tinyurl.com/DEMCFeedback

i4talent DATA SCIENCE