

goto;

# GOTO AARHUS 2022

#GOTOaar







# Bridging the chasm between Research and Software Development

A story from a fertility app

goto aarhus, 15 June 2022

Linda Stougaard Nielsen, Director Data Science, Ava AG





Linda Stougaard  
Nielsen, Ph.D.

Director Data Science  
Ava AG

## Agenda

Presenting ava – vision and products

The problem – two examples

Possible solutions

Remaining challenges

Q&A





# Ava AG

Swiss company

Founded 2014

Awards in startup, medtech, femtech

Wearable device collecting signals:

- Heart rate, heart rate variability, temperature, breathing rate, perfusion, sleep state every 10 s
- 200'000 users each collecting 1 mio data points every night

Applications:

- Fertility app launched in 2016 (CE & FDA approved)
- Contraception app launching 2023 in Europe
- Side track: Covid early detection (research project)



Ava’s vision is to assist women throughout their life with their reproductive health

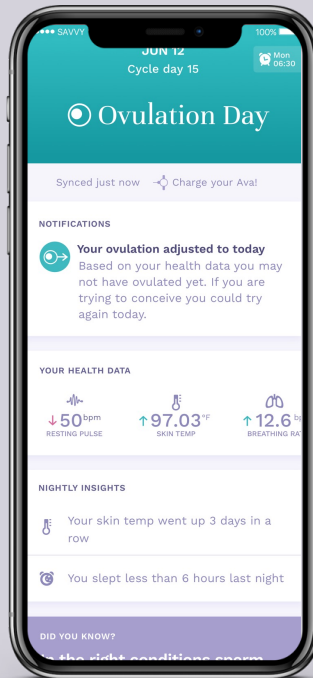


Track Cycle



Helps young adults make sense of their bodies and cycle

Conception



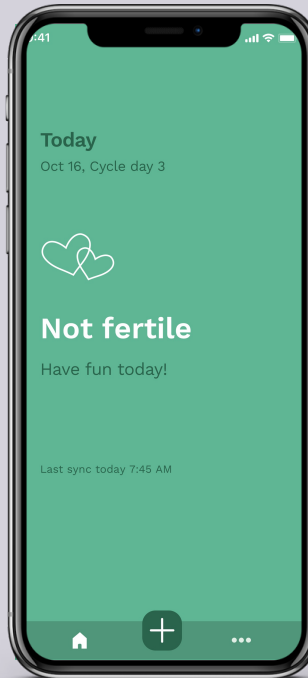
Doubles chances of pregnancy

Pregnancy



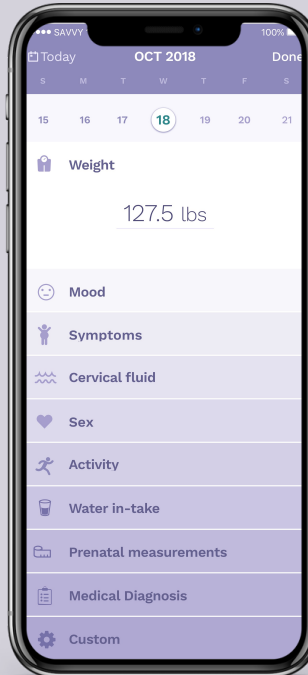
Monitors for healthy pregnancy

Contraception



Fully certified digital birth control (in progress)

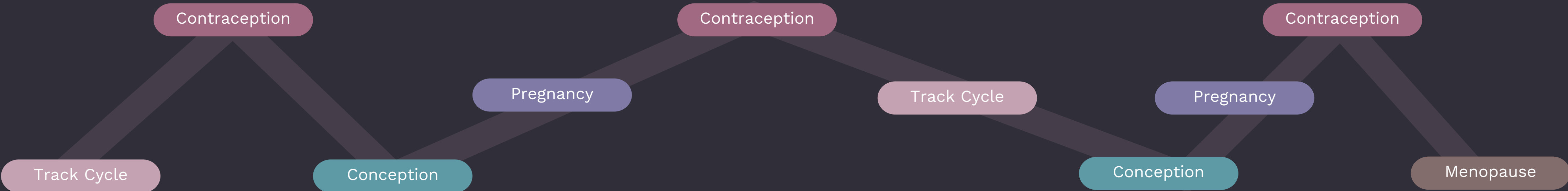
Menopause



Monitors for healthy peri-menopausal life (not started)

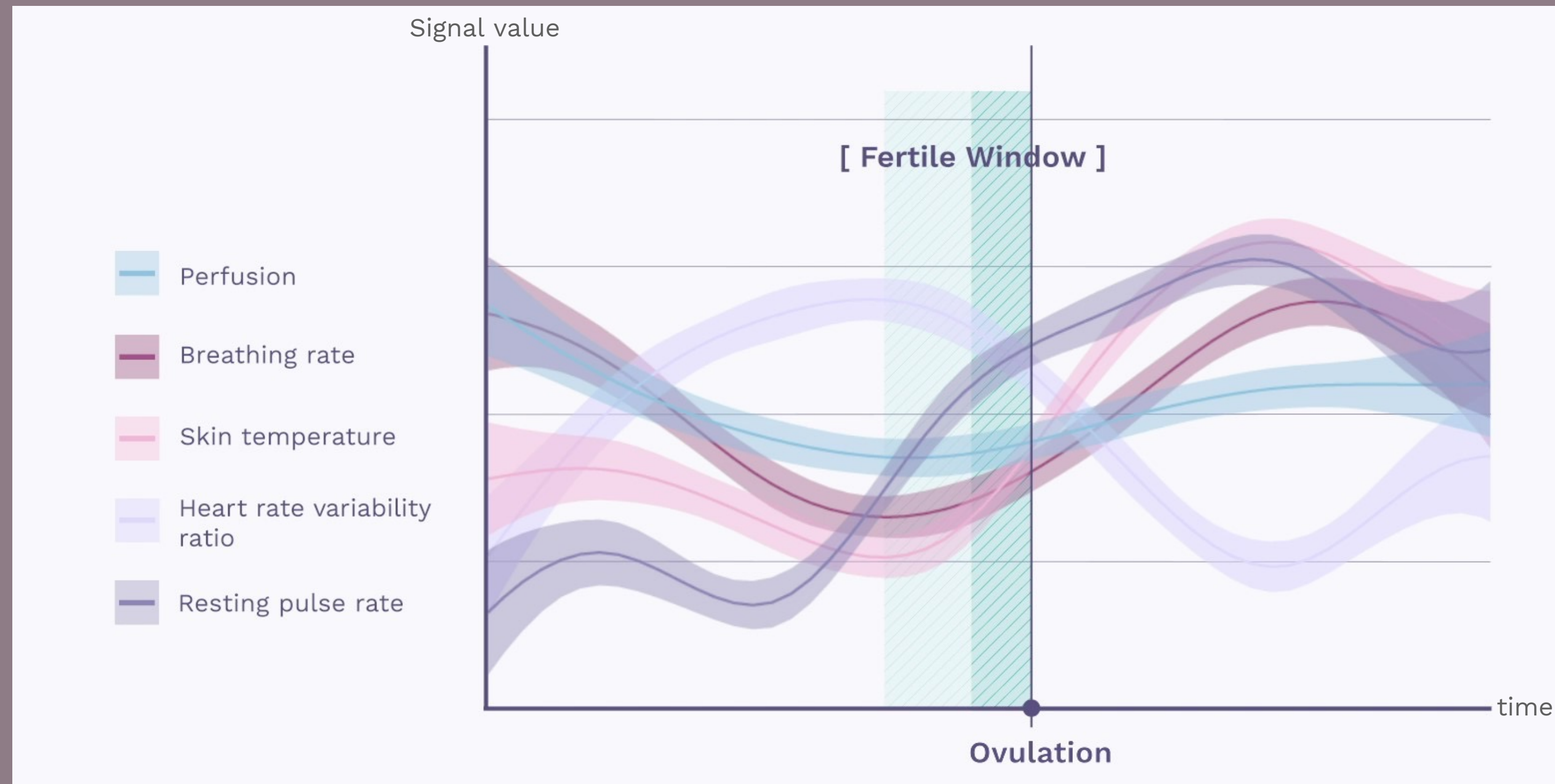
ava fertility app

ava prevent app





## How does it work



*Wearable Sensors Reveal Menses-driven Changes in Physiology and Enable Prediction of the Fertile Window: an Observational Study*, Goodale, B. M., Shilaih, M., Falco, L., Dammeier, F., Hamvas, G., & Leeners, B. (2019)



## Medical research

- Hormonal changes throughout cycle, pregnancy, infections, menopause
- Influence on physiological signals

## Clinical trials

- Cooperation with research institutes
- Certification and postmarket surveillance
- Statistical analyses

## Machine learning

- Signal processing
- Classical statistics and machine-learning
- Neural Networks
- Constraints on noisy data and risk





“One of the hardest parts of machine learning is effectively putting models in production.”

<https://neptune.ai>



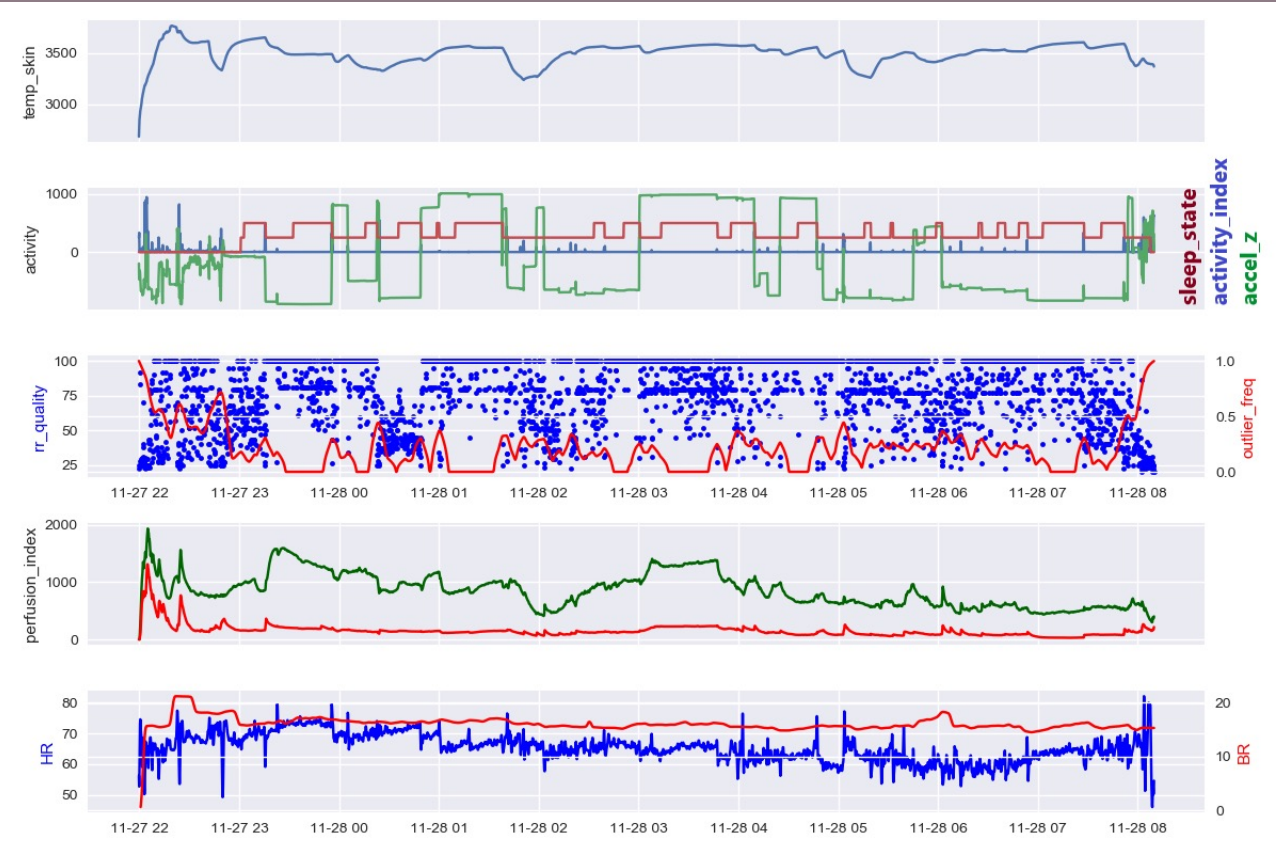
# The problem

2 examples





# Example 1: re-process all raw data files



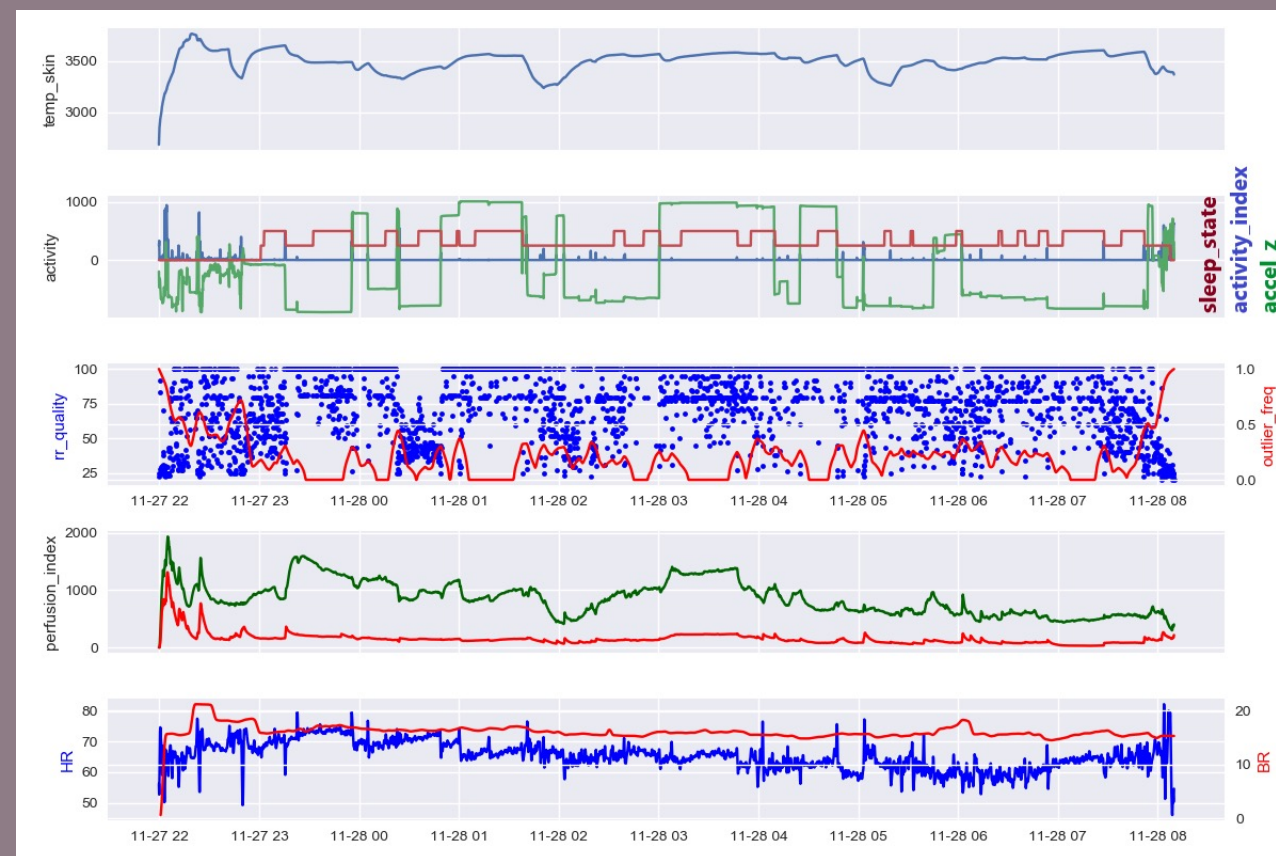
Parse, split, filter  
Extract nightly features  
Assess quality

	EXISTING
Production code	Python module for data processing
Scaling solution	Spark on AWS
Challenges	<ul style="list-style-type: none"><li>- Spark own language</li><li>- Re-implement solution</li><li>- Maintain 2 parallel implementations</li></ul>





## Example 1: re-process all raw data files



Parse, split, filter  
Extract nightly features  
Assess quality

	OLD	NEW
Production code	Python module for data processing	
Scaling solution	Spark on AWS	Parallel on AWS
Challenges	<ul style="list-style-type: none"><li>- Spark own language</li><li>- <b>Re-implement solution</b></li><li>- Maintain 2 parallel implementations</li></ul>	<ul style="list-style-type: none"><li>- Additional implementation of infrastructure to run parallelisation</li></ul>



## Example 2: Re-train a tensorflow model

Tensorflow model – not the issue (ok packaged)

Data handling:

- Input data into tensor
- Normalisation
- Filtering of data (outlier detection)
- Model output into app input

Scattered all over the code

- Impossible to follow and to maintain / extend

Different code than used in training

- Impossible to prove consistency between eg. outlier handling in training code vs. production code

### Well-known issue a.k.a “glue code”

- supporting code for getting data in and out of generic ML packages
- ~95% of code
- anti-pattern

### Re-implementation of code





## Issues identified

- Re-implementation of code
- “Glue code”



# Solutions

Just a question of writing good code, right?



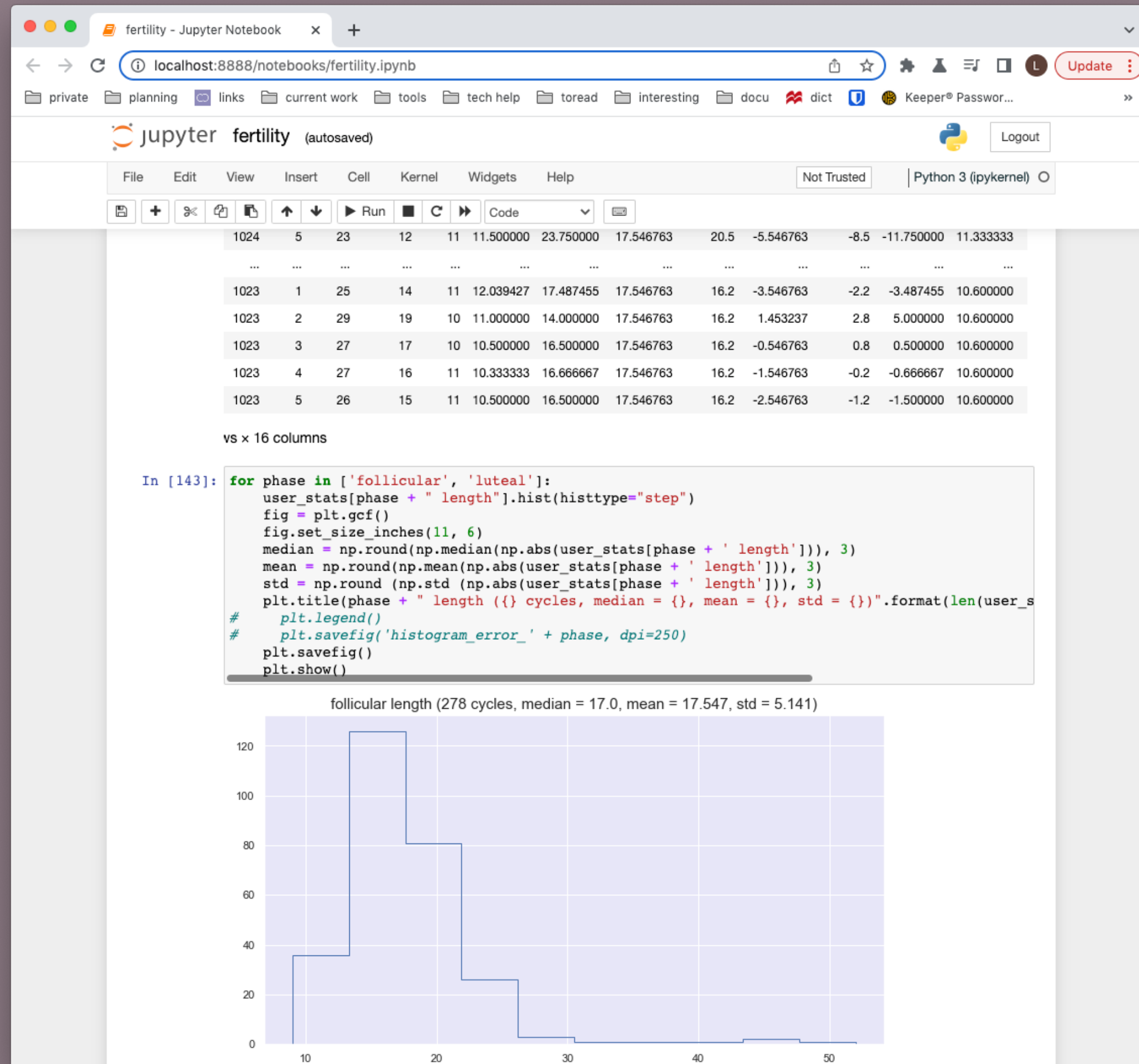
## Re-use of code

- But some-one has to write the code that can be re-used
- And it may require additional infrastructure (eg. batch processing)
- And what about experimentation: modifying code slightly in various ways





# Experimentation / research



## Research code

### Scientific tools are different

- Ex Jupyter Notebook
- Line-by-line execution

### Iterative approach

- Experimentation requires trial & error
- Lot of alternative code for different approaches
- Often incorporates with external tools (ex tensorboard for training)

### Scientists are generalists

- “One solution to fits all”  
-> End up with long scripts that can do EVERYTHING
- Not specific to single problem (many trials)





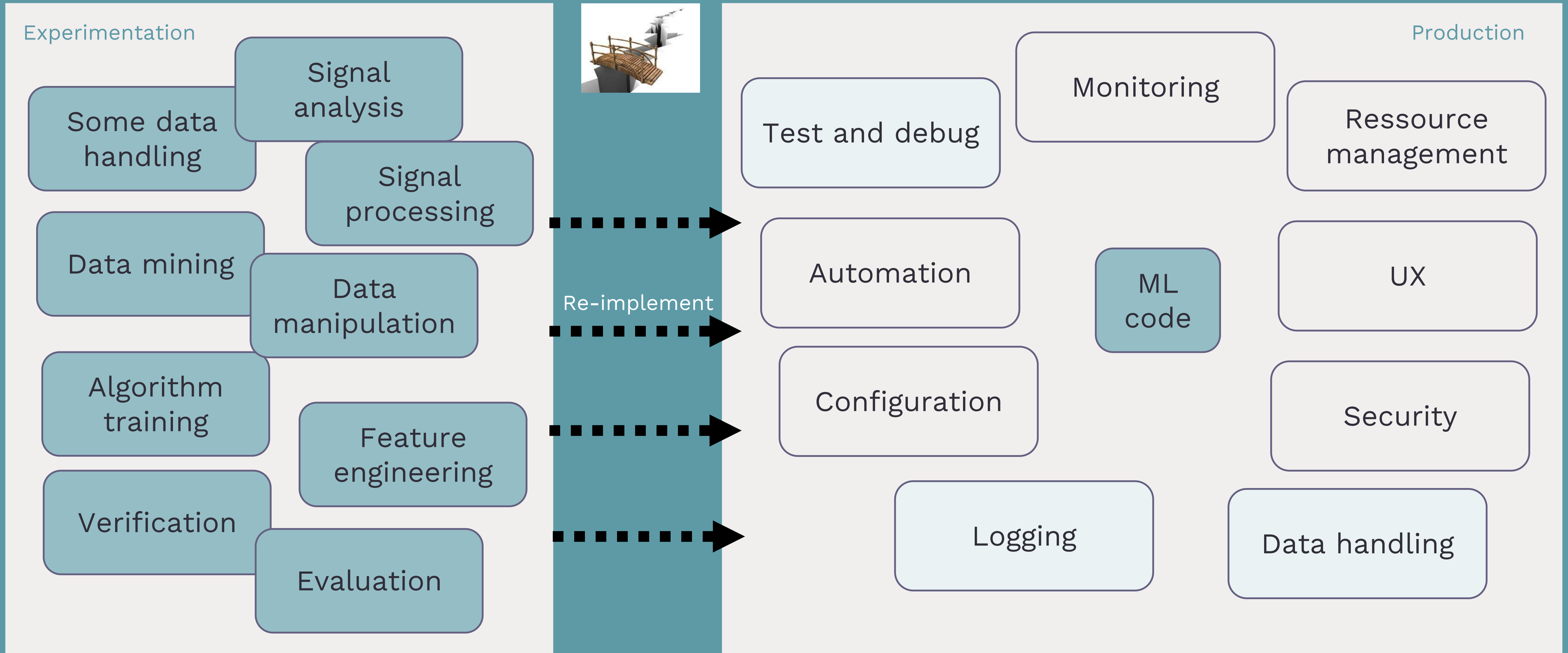
## Transitioning from research code to production code

Research code	➔ Production code	Consequences / differences
Flexibility (can do everything)	Specialised (one purpose / fixed final version)	Full of redundant code
Manual usage	Automated repeated usage	No focus on stability, exception handling, and unit testing
Used by 1 person	Maintained by many people	Not following best practices
Row by row execution	Script execution	Not object oriented / patterns
Stand-alone	Interacting with system	No clear interfaces
Data from large data sets	One data point from stream	Data handling is different
Need parallelising within	Need parallelising around	Need restructuring and optimising





# Machine learning code into production (bridging the chasm)







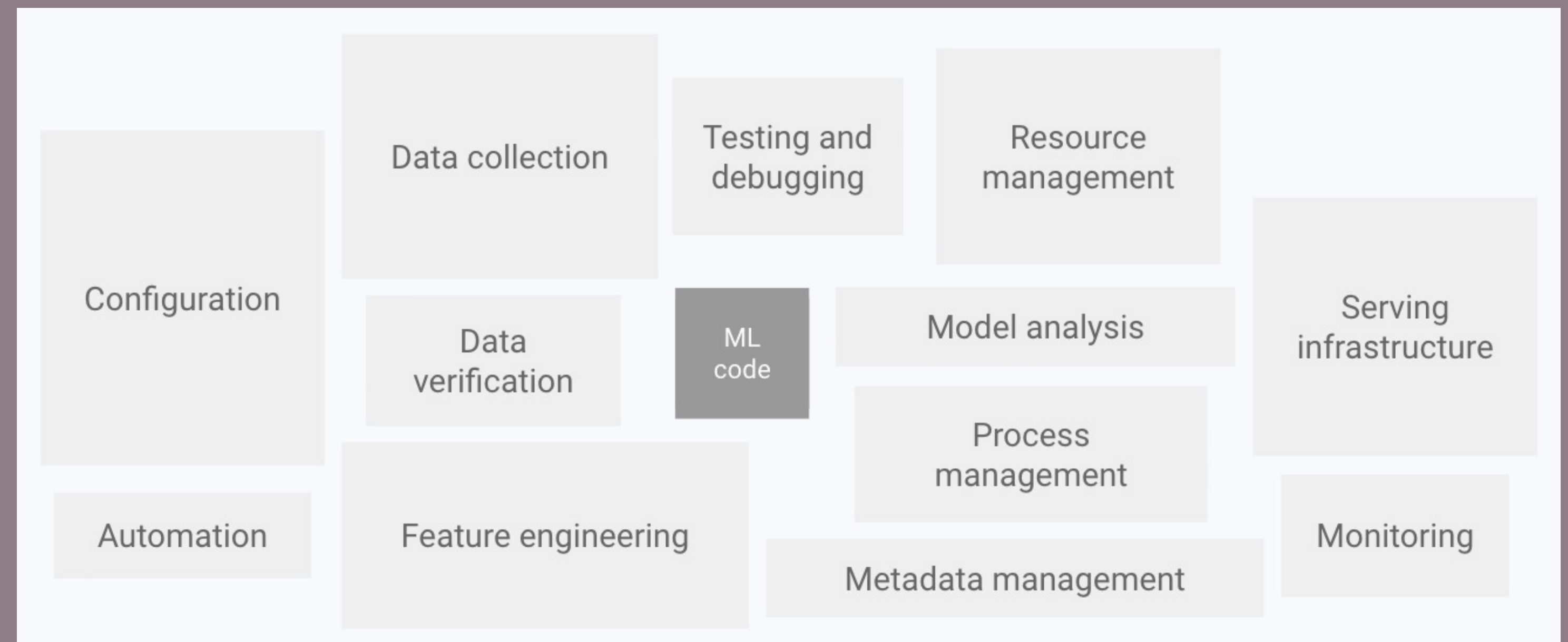
## Required infrastructure surrounding lifecycle of ML code

### Glue code (ML):

- Getting data in and out of ML packages

### Glue code (SW):

- Connecting incompatible components
- No contribution towards requirements



Source: “Hidden Technical Debt in Machine Learning Systems”, NIPS 28 (2015) by Google





## Who should implement all this glue code

Educate researchers to be good SW devs

NO!

- No interest / no skills in this area
- Focus on research / modelling / data processing

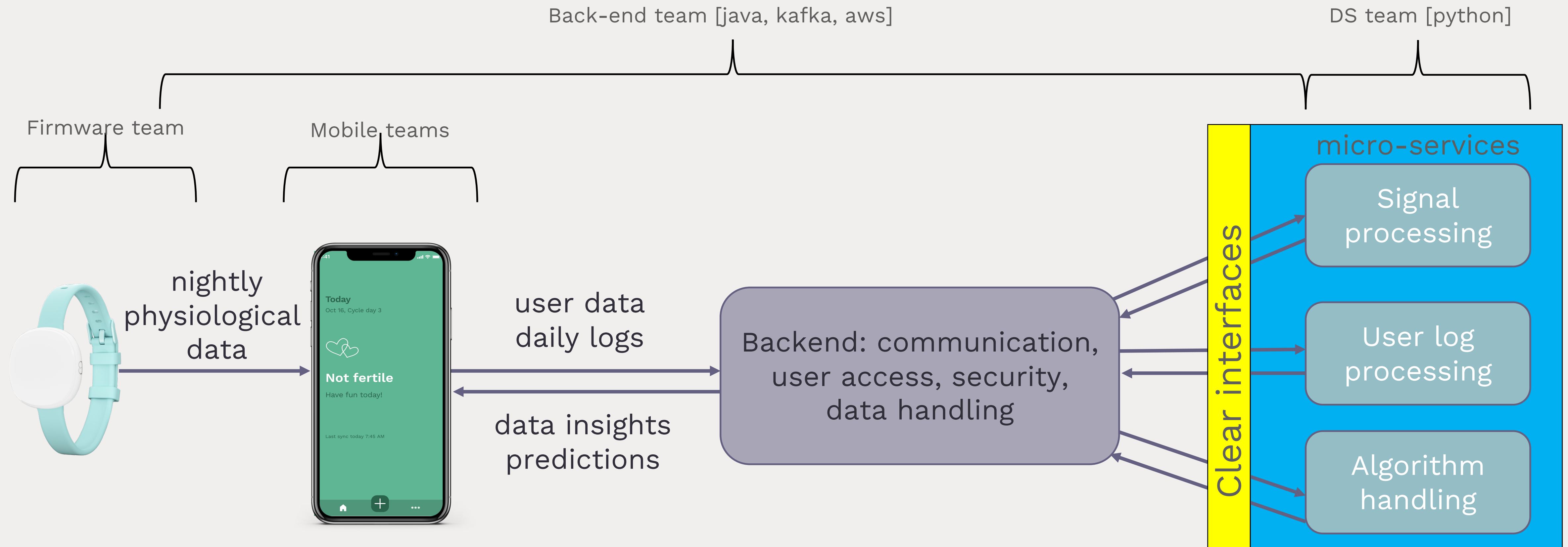
Let SW devs build the code

Depends...

- Parallel implementations
- Inconsistency problems
- Time to market



# High level architecture

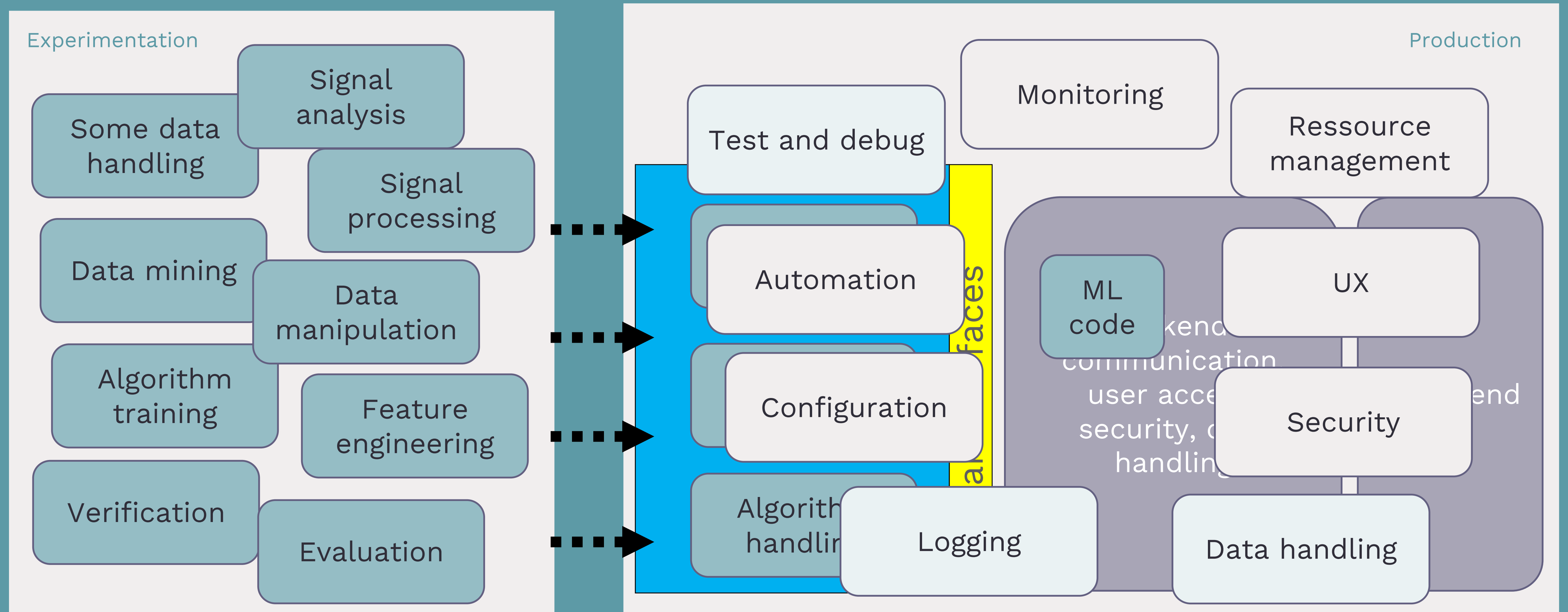


devops: infrastructure, integration, code versioning, automation, scaling, monitoring





# ML modules as microservices

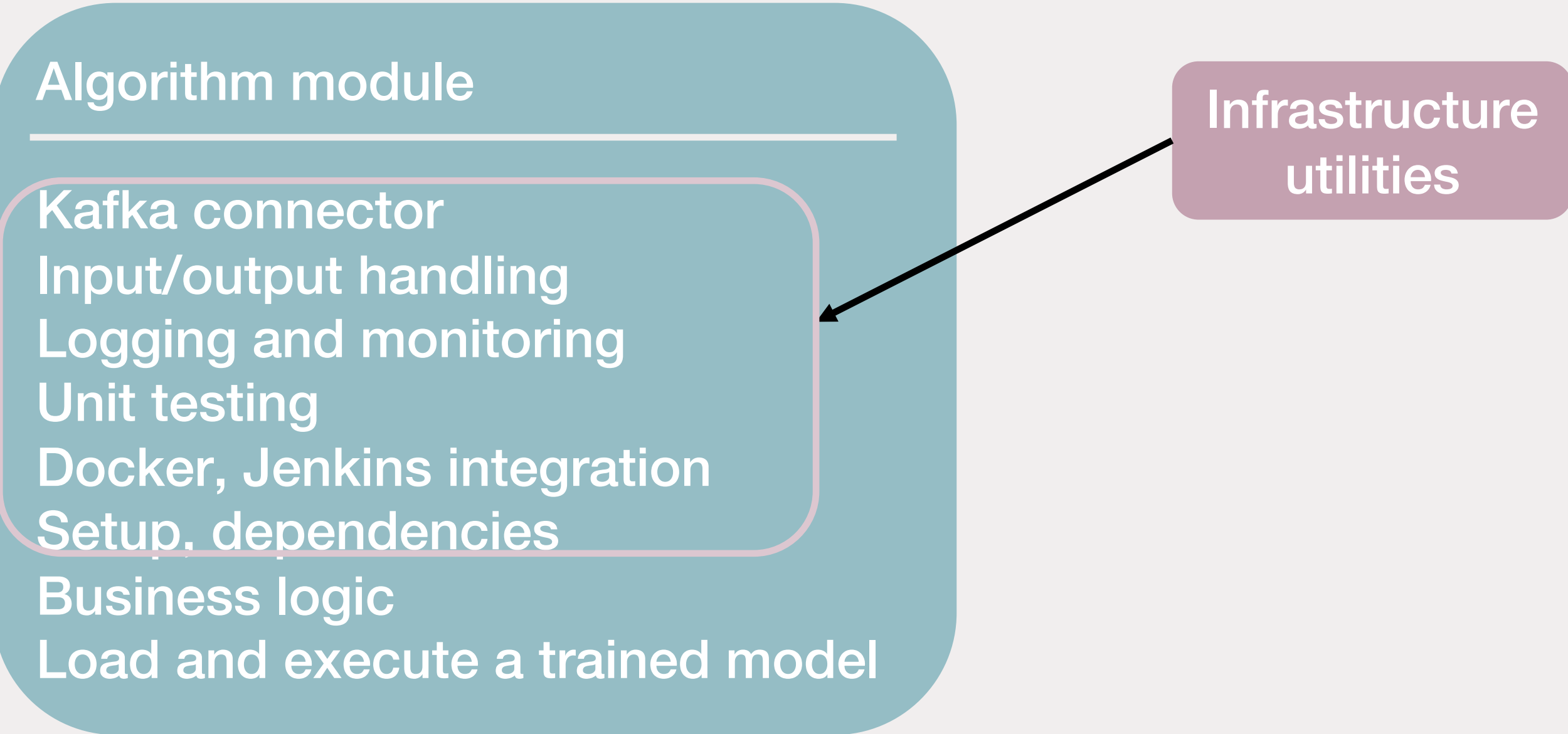


devops: infrastructure, integration, code versioning, automation, scaling, monitoring





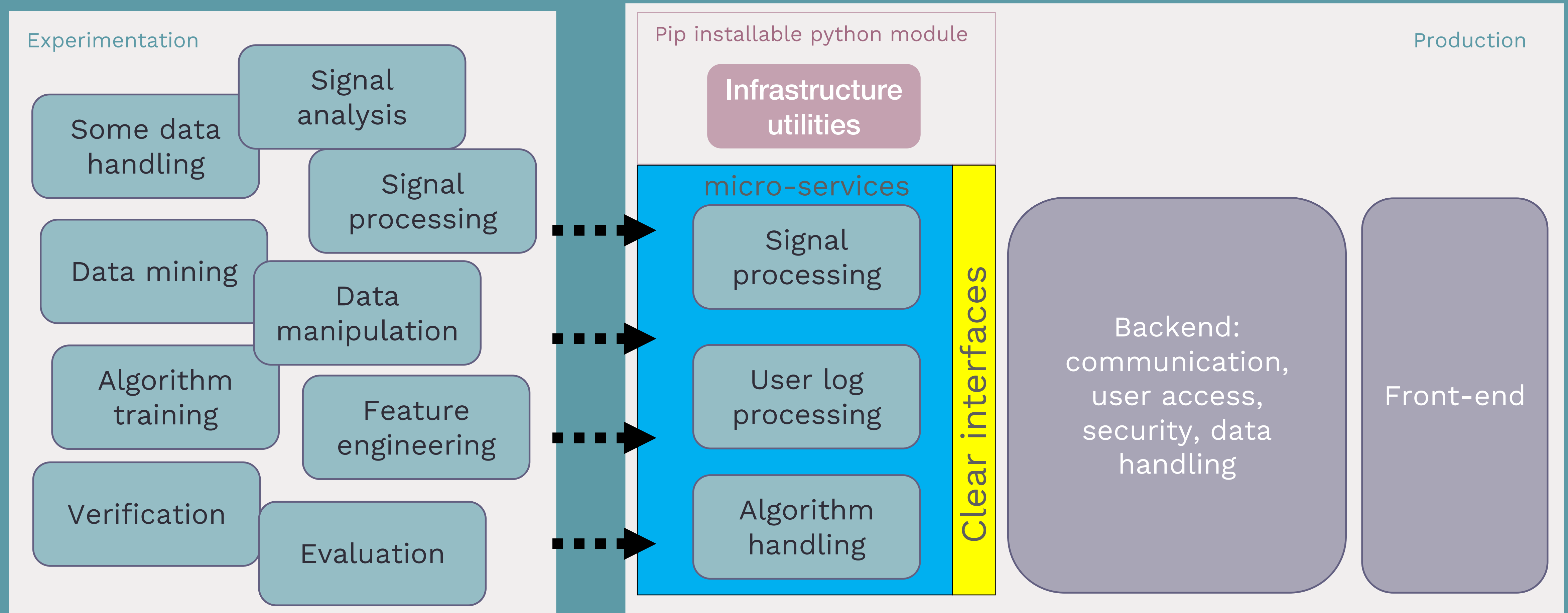
# Modularise the DS modules







# Microservices and modularising the common code

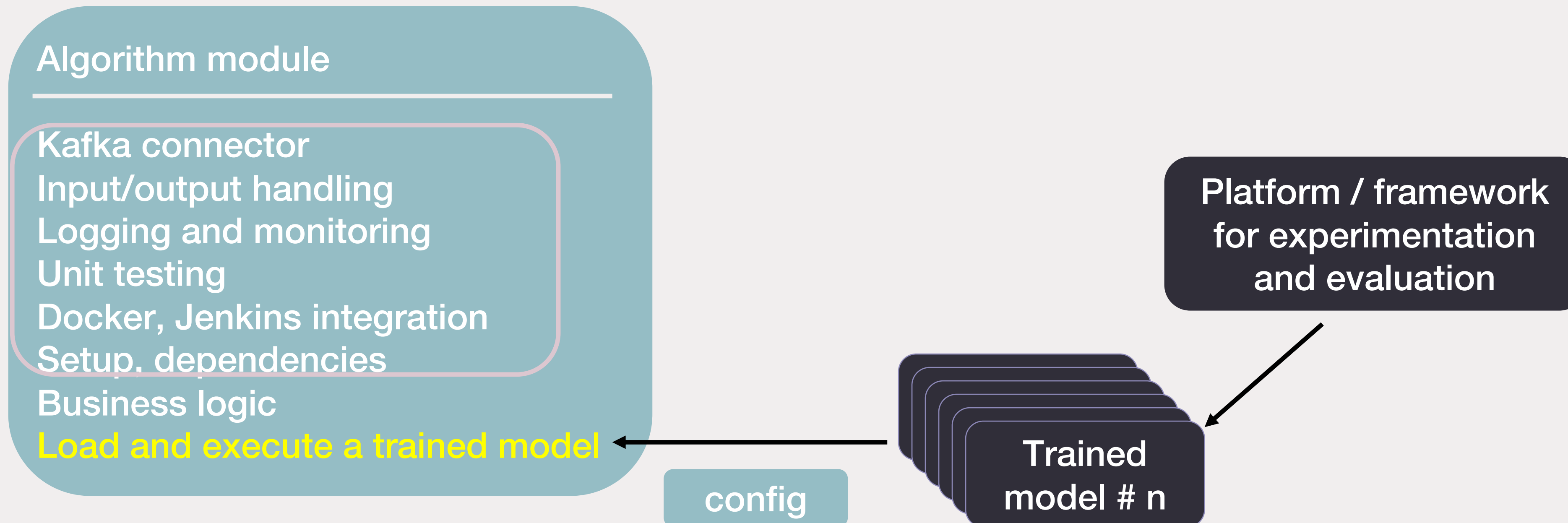


devops: infrastructure, integration, code versioning, automation, scaling, monitoring







## Inject trained models

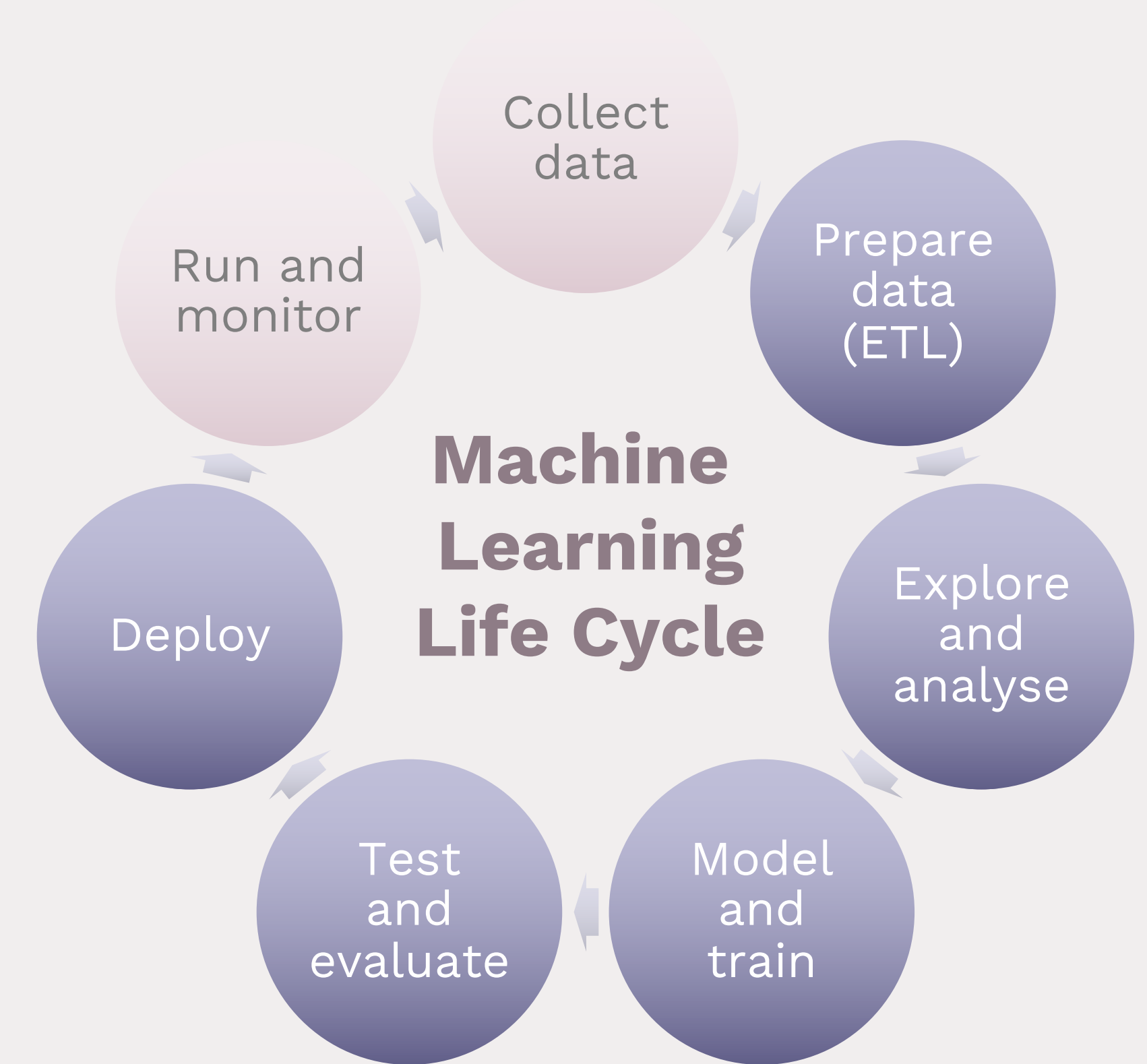






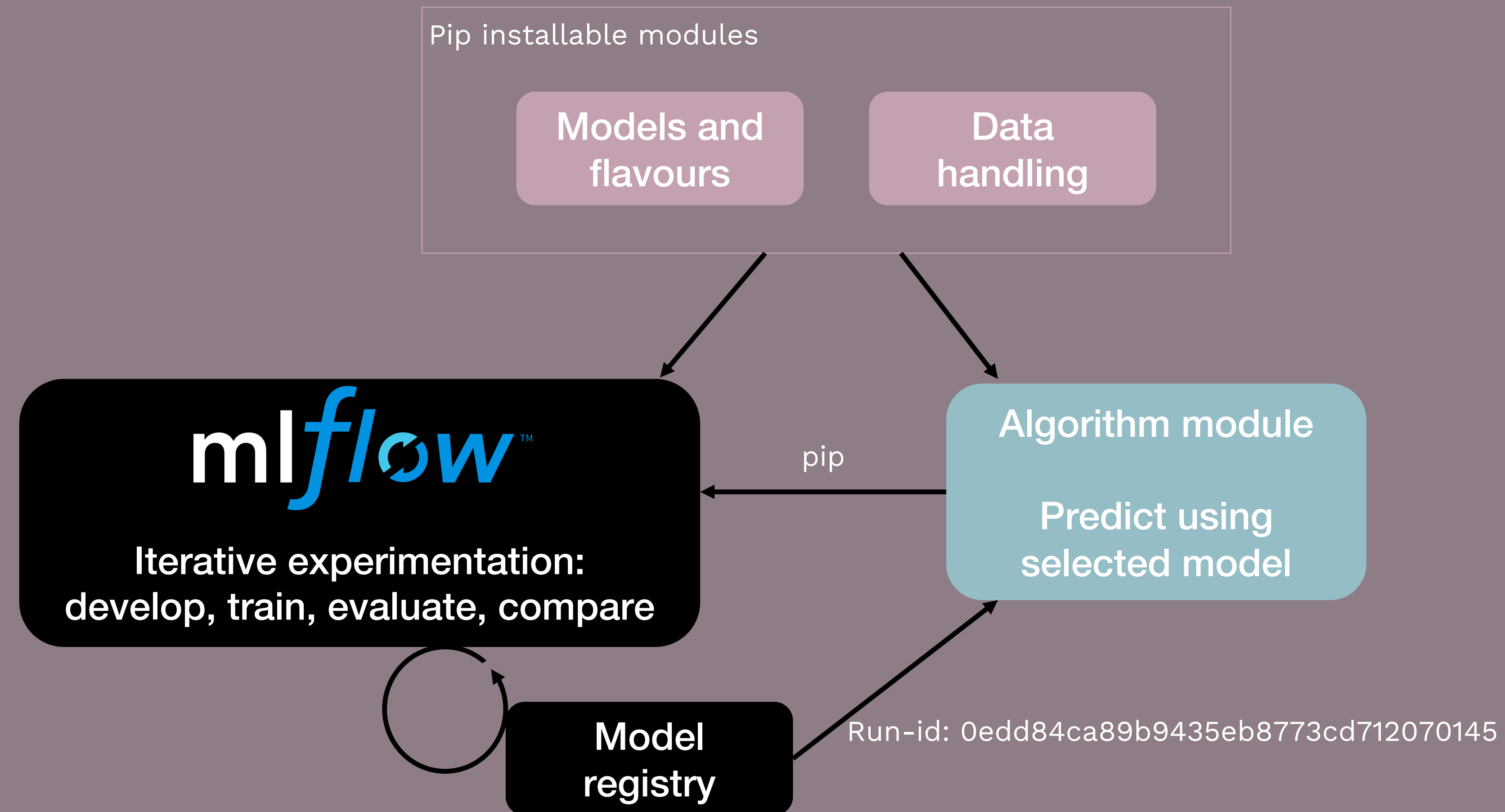
Most popular mlops tools for managing ML life cycle

 <b>Kubeflow</b>	
Google	Databricks
Tensorflow (TFX)	Python
Pipeline based	Experimentation based
Deploy and manage complex ML systems	Tracking of experiments
Kubernetes infrastructure (more complex)	Local or cloud, eg develop locally and track remote





# ML lifecycle Architecture





Experiments

+

Search Experiments

Default

my\_exp

pipeline1

prepare\_data

train\_pipe\_0613

train\_pipe\_0613

Track machine learning training runs in an experiment. [Learn more](#)

Experiment ID: 4

Description

Edit

Refresh

Compare

Delete

Download CSV

Start Time

All time

Columns

Only show differences

metrics.rmse < 1 and params.model = "tree"

Search

Filter

Clear

Showing 8 matching runs

								Metrics		Parameters
	Start Time	Duration	Run	User	Source	Version	Models	close score	closed cyc	cli ea mi mean_thr score threshold model n_iter run_id
	24 minutes	6.1 min	-	linda	train_pipel	117df2	lowess-gri.../4	-2.597	0.35	( 0.366 -2.629 0.175 se... 20.0 4db38753...
	3 hours agc	4.8min	-	linda	train_pipel	b4dfb3	lowess-gri.../3	-	0.361	- 0.359 -2.652 0.185 se... 20.0 4db38753...
	3 hours agc	4.4min	-	linda	train_pipel	b4dfb3	lowess-gri.../2	-	0.344	- 0.364 -2.629 0.243 se... 10.0 4db38753...
	4 hours agc	4.5min	-	linda	train_pipel	b4dfb3	lowess-gri.../1	-	0.362	- 0.358 -2.657 0.186 se... 10.0 4db38753...
	4 hours agc	21.7s	-	linda	train_pipel	b4dfb3	lowess-gri.../22	-	0.333	- 0.38 -3.167 0.194 se... - 08629a84...
	4 hours agc	11.6s	-	linda	train_pipel	1b6957	lowess-gri.../20	-	0.556	- 0.257 -2.222 0.181 se... - 08629a84...
	5 hours agc	8.8s	-	linda	train_pipel	1b6957	lowess-gri.../19	-	0.444	- 0.291 -2.444 0.194 se... - 08629a84...
	5 hours agc	8.1s	-	linda	train_pipel	1b6957	lowess-gri.../18	-	0.444	- 0.298 -2.444 0.173 se... - 08629a84...

# Experimentation

Try out different models and data sets  
Track: parameters, metrics, and artifacts  
Comparison between experiments

# Model registry

Store trained models for later use  
Load selected trained model by ID

train\_pipe\_0613

Run fe838e59bfc149daa4a83960fc0c465

Date: 2022-06-13 21:41:43

Source: train\_pipeline.py

Git Commit: 117df2e2b0d5c0

User: linda

Duration: 6.1min

Status: FINISHED

Lifecycle Stage: active

Description

Edit

Parameters (3)

Metrics (8)

Tags

Artifacts

closing-model-pipeline

mlmodel

conda.yaml

model.pkl

requirements.txt

ovv+1\_distance\_distribution.png

raw\_scores.csv

raw\_scores.txt

Full Path: /minusrn/4/fe838e59bfc149daa4a83960fc0c465/artifacts/ovv+1\_distance\_distribution.png

Size: 18.28KB

Distribution of closure distances

Registered Models

lowess-gridsearch-aacm

lowess-gridsearch-aacm

Created Time: 2022-06-13 17:46:55

Last Modified: 2022-06-13 21:47:50

Description

Edit

Tags

Versions

All

Active 0

Compare

Version	Registered at
Version 4	2022-06-13 21:47:50
Version 3	2022-06-13 18:47:38
Version 2	2022-06-13 18:19:06
Version 1	2022-06-13 17:46:55

# Advantages

Tracking of experiments  
Reproducibility (models, params, data sets)  
Easy deployment of trained models





Everything needed for serving model is automatically saved

Project

closing\_algo

data

datahandler

evaluate\_algoworker

mlruns

.trash

0

0d2bd35b61b74f0da443838ed64becc4

0edd84ca89b9435eb8773cd712070145

artifacts

closing-model

conda.yaml

MLmodel

model.pkl

requirements.txt

0fbbff4079dc47b5a95ba74b0c2b9f84

1b3ede8c3f544abcad32686ba5b1a426

1c84b30255de476b9ab8b5b44369e4d6

1d537fbb1ab34da28bbd549c09d5b69f

0001ea1ebb9e43fa9bfed912a1445e41

README.md

train.py

mlruns/.../conda.yaml

False

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

artifact\_path: closing-model

flavors:

python\_function:

env: conda.yaml

loader\_module: mlflow.sklearn

model\_path: model.pkl

python\_version: 3.7.11

sklearn:

pickled\_model: model.pkl

serialization\_format: cloudpickle

sklearn\_version: 1.0.2

model\_uuid: 2552f3fc84d14f9fbd3c7df4f198adea

run\_id: 0edd84ca89b9435eb8773cd712070145

utc\_time\_created: '2022-03-22 17:17:04.702535'

© Ava 2022 | Advancing women's health

28





Follow pattern from *scikit-learn* (*sklearn*)

### Standard classes eg:

- RandomForestClassifier
- BaggingClassifier
- GridSearchCV
- StandardScaler
- Pipeline

```
class FertilityModel(BaseEstimator, ClassifierMixin, metaclass=abc.ABCMeta):
    MODEL = "algo_model"
    MODEL_PARAMETERS = "algo_model_parameters"

    def __init__(self):...

    @abc.abstractmethod
    def fit(self, X=None, y=None, sample_weight=None, check_input=True):...

    @abc.abstractmethod
    def predict(self, cycle: CycleData, *args, **kwargs) -> (np.ndarray, np.ndarray):...

    @classmethod
    def get_model_name(cls):...

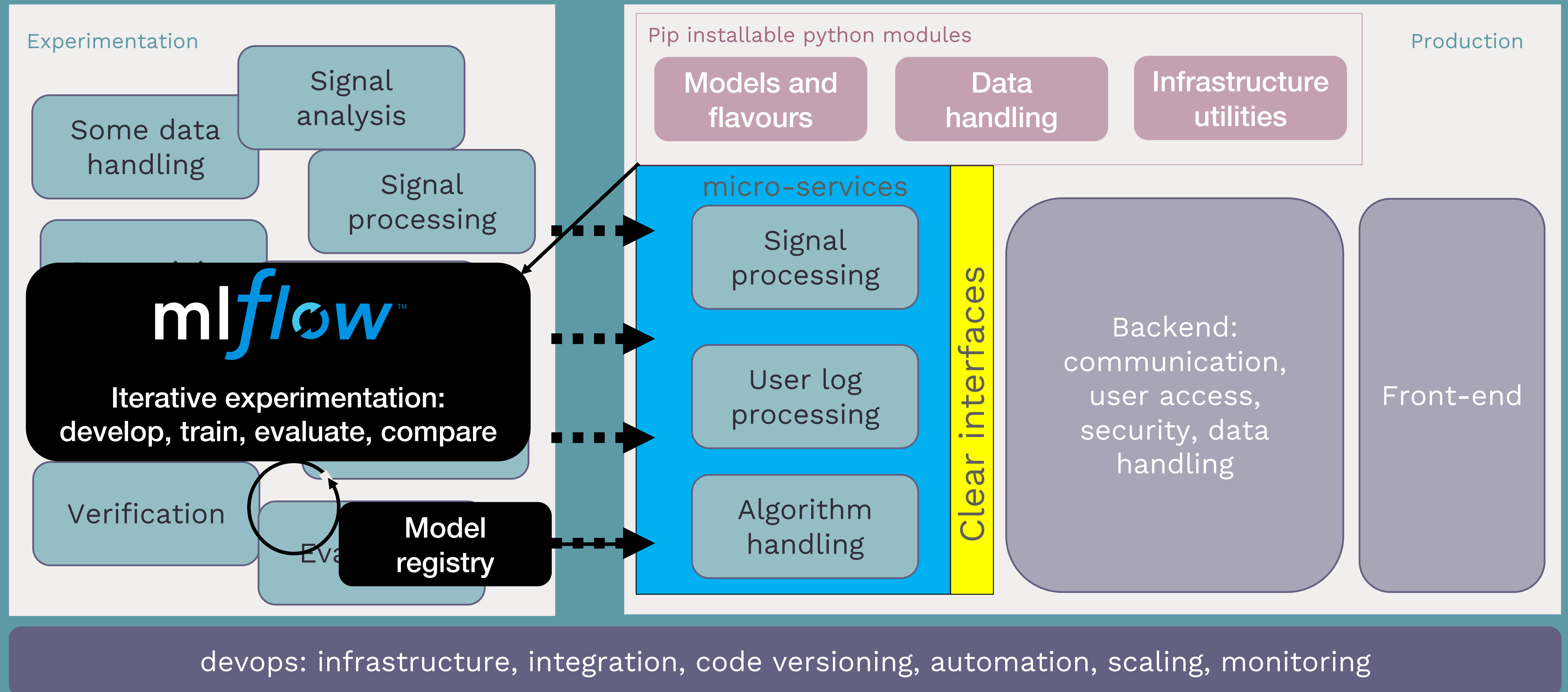
    def serialise(self) -> dict:...

    def score(
        self, cycle: CycleData, true_fertility_indications, sample_weight=None
    ) -> float:...
```





# Mlflow for experimentation and model serving





# Summary

## And challenges



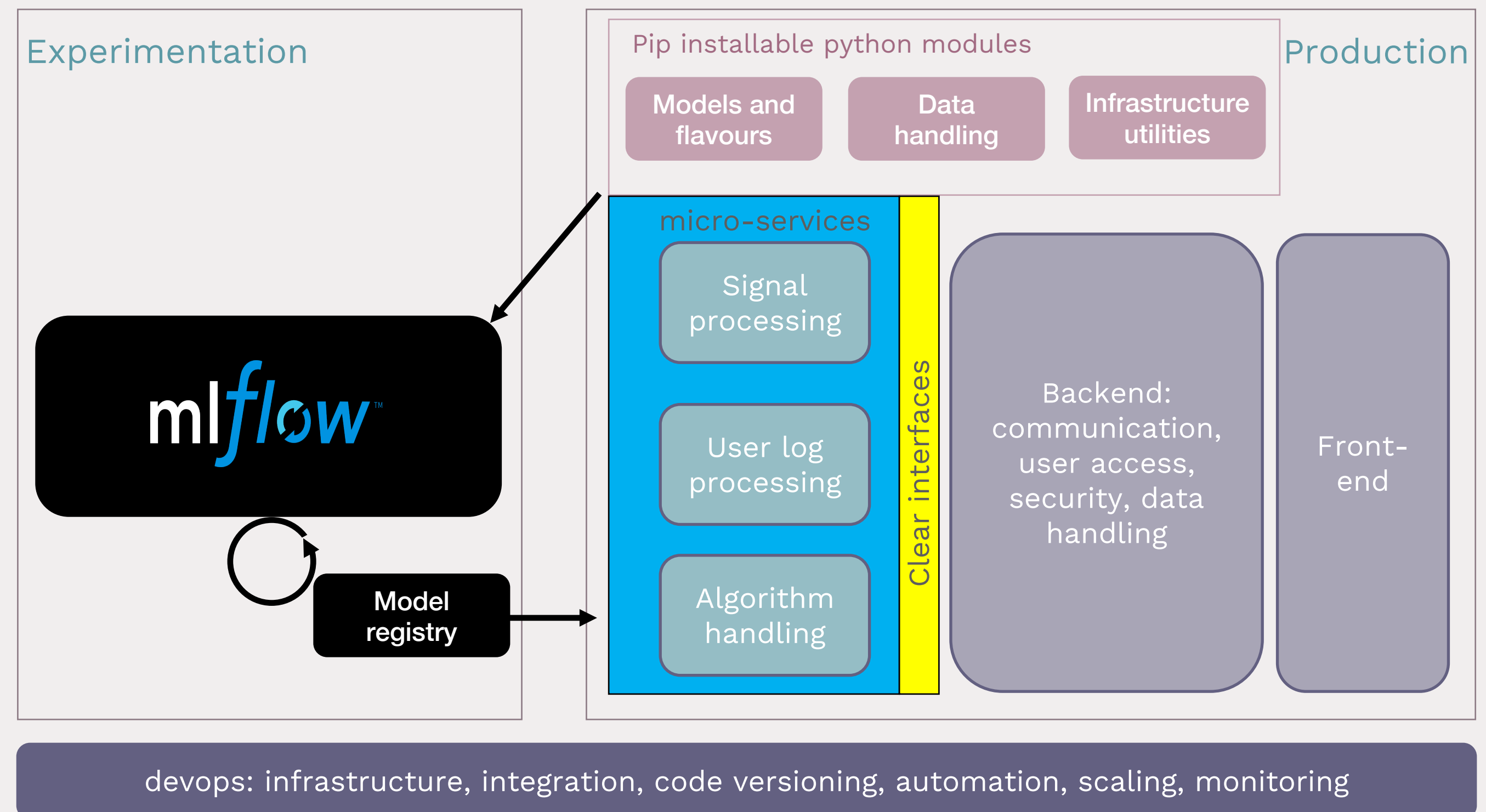


## Solution at ava

Separate data science modules as micro-services with clean interfaces to the rest of the system (most of the glue)

Separate parts of each data science module that is related to standard software tasks (some of the glue)

The core part is the models, the lifecycle is managed by a mlops tool and a trained model is stored in a blob





## Remaining challenges

It is still possible to build a mess

- Follow best practices
- Recognize when code can be re-used
- Communication to prevent duplication

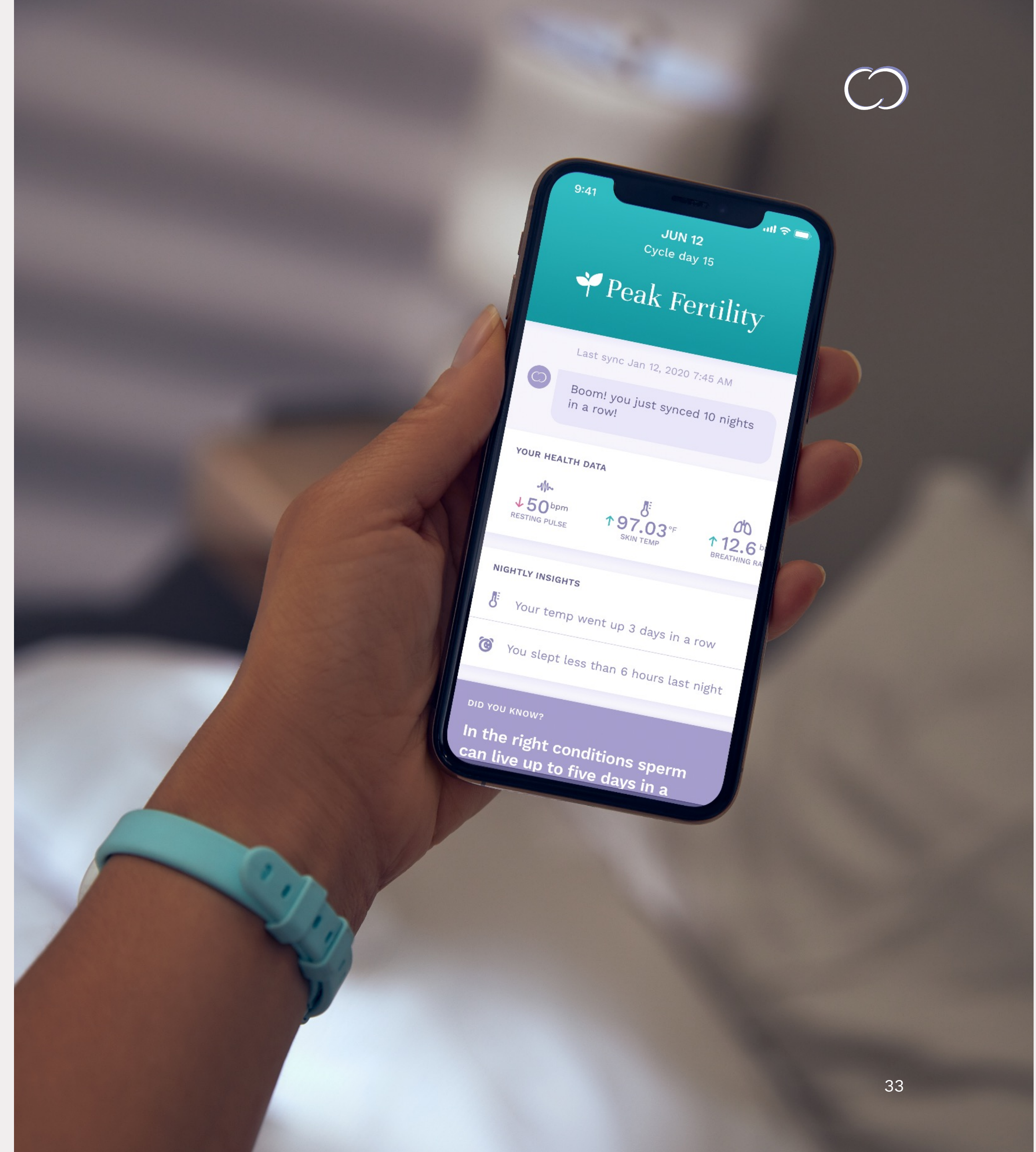
It is out of comfort zone for most researchers

- Different tools
- Outside core competences
- Need training and support
- Still some researchers will never go down this path...

ML engineer / SW developer is needed for parts

- To write the glue
- Use same libraries and language
- Coordination and communication

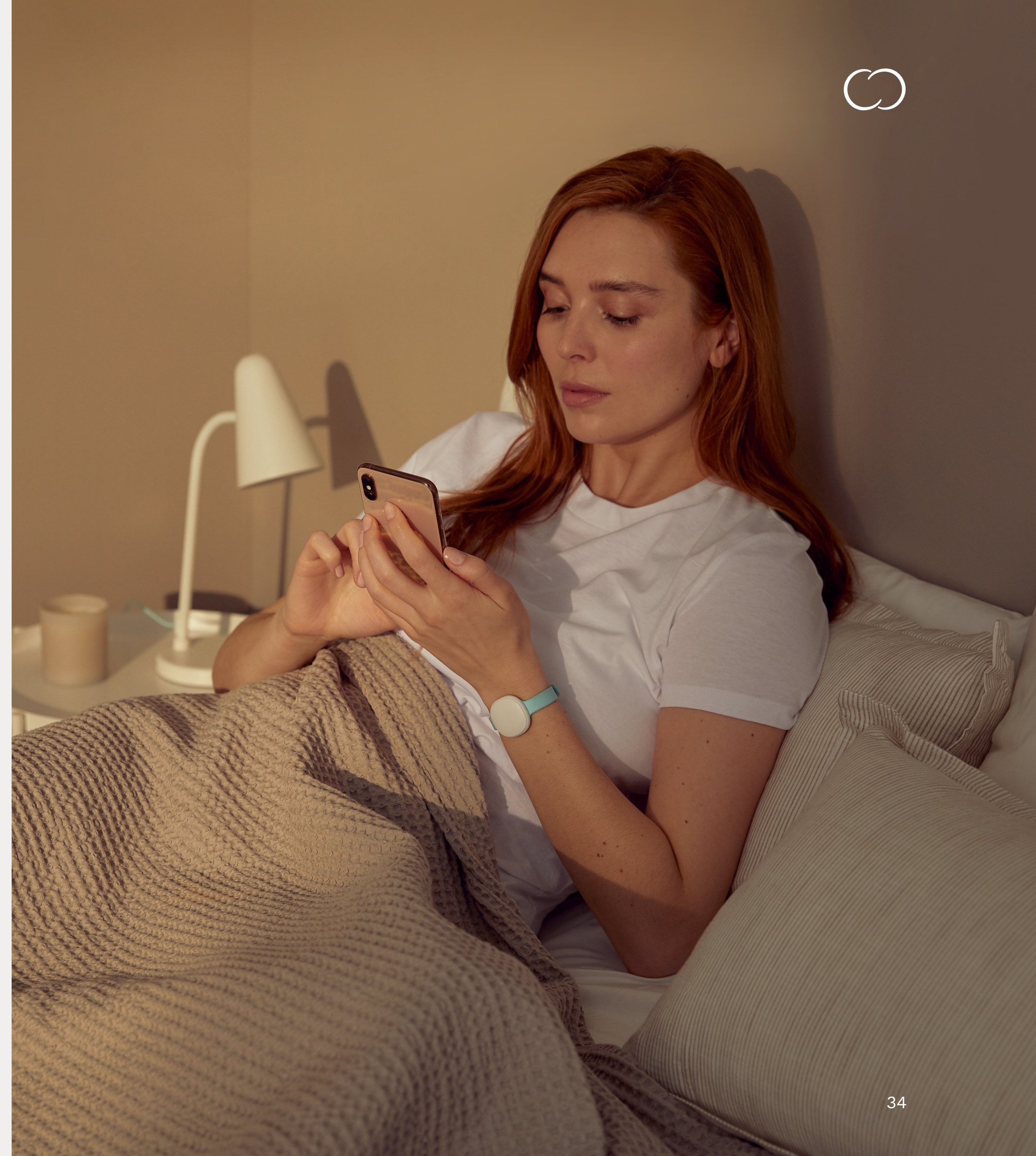
Not solving all problems (eg. Example 1)





– and the solution is agile?

- ✓ Cross-functional teams:  
researchers and data engineers / sw devs  
T-shaped skills
- ✓ Pairing up, code review
- ✓ Work on different projects







Thank you!

Questions?



goto;

# DON'T FORGET TO **RATE THE SESSIONS**

#GOTOaar

Rate a minimum of **5 sessions** and  
claim your **reward** at the  
Registration Desk at the Trifork Hall