

femtec health avacō

Bridging the chasm between Research and Software Development

By Linda Stougaard Nielsen
At goto copenhagen, 4 October 2022



Linda Stougaard
Nielsen, Ph.D.

Director Data Science
Femtec health

Agenda

Context – company & products

The problem – two examples

Possible solutions

Remaining challenges

Q&A



ava – a femtec health company

Swiss company, 2014, awards in startup, medtec

Acquired by femtec health 2022

Female reproductive health

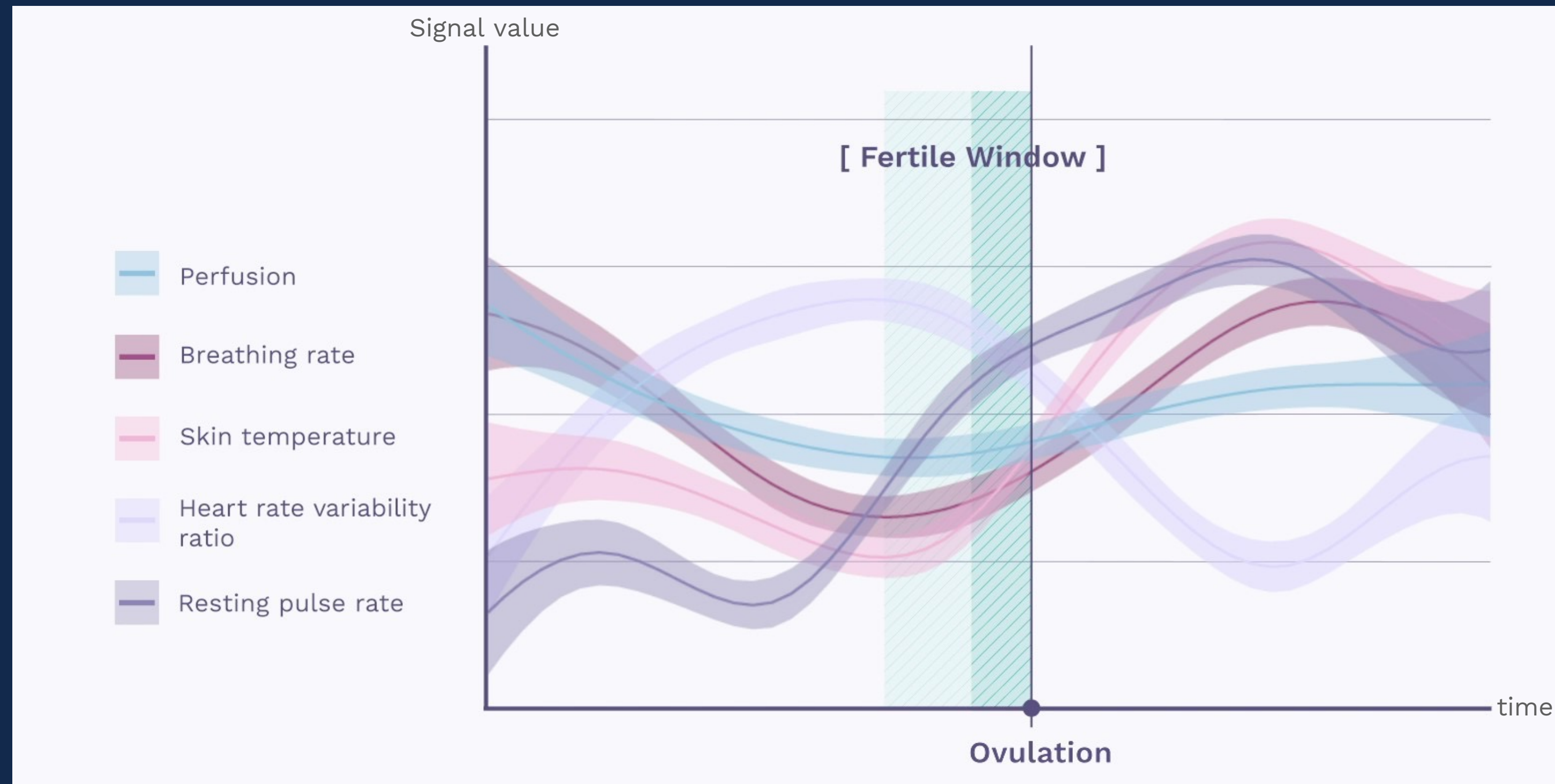
Wearable device collecting signals:

- 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
- Covid early detection (research project)

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.”

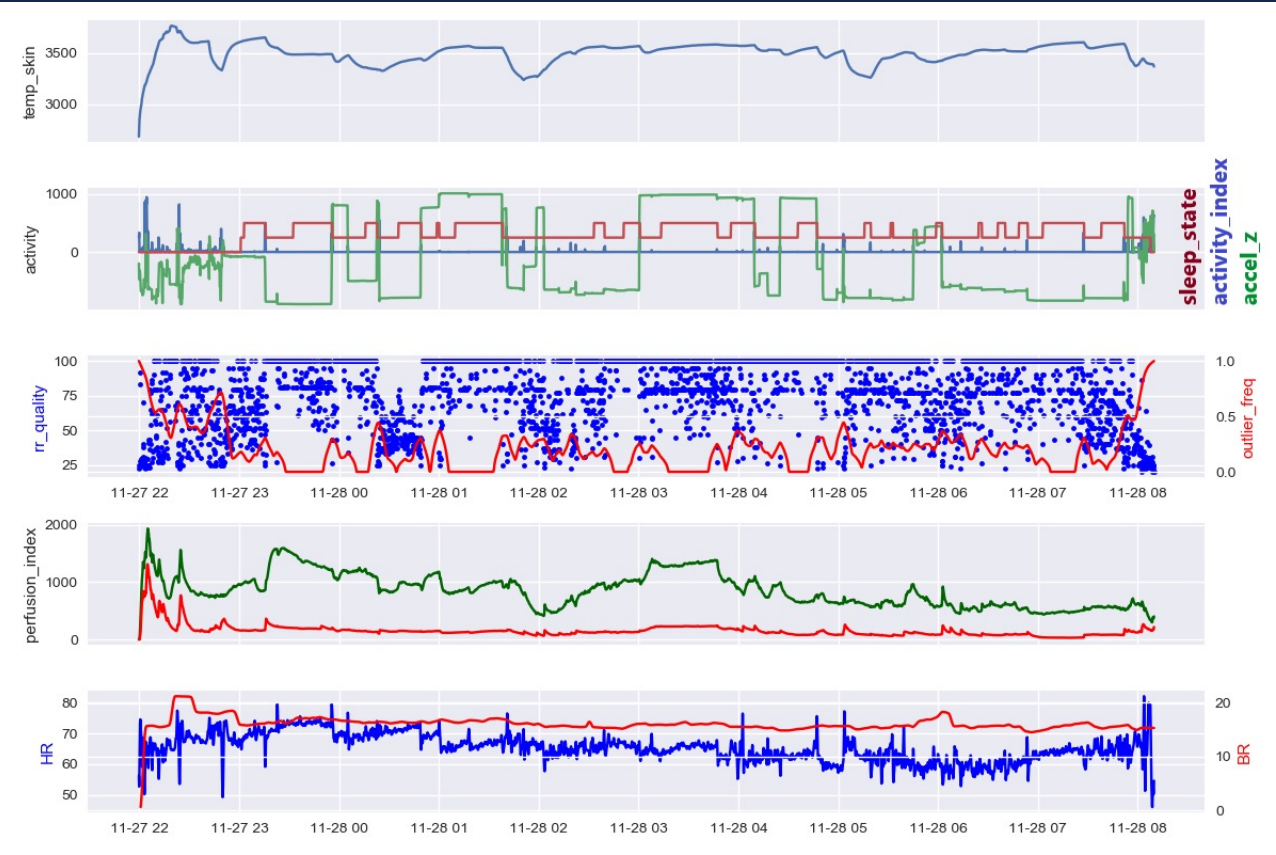
<https://neptune.ai>

The problem

2 examples



Example 1: Re-process all raw data files

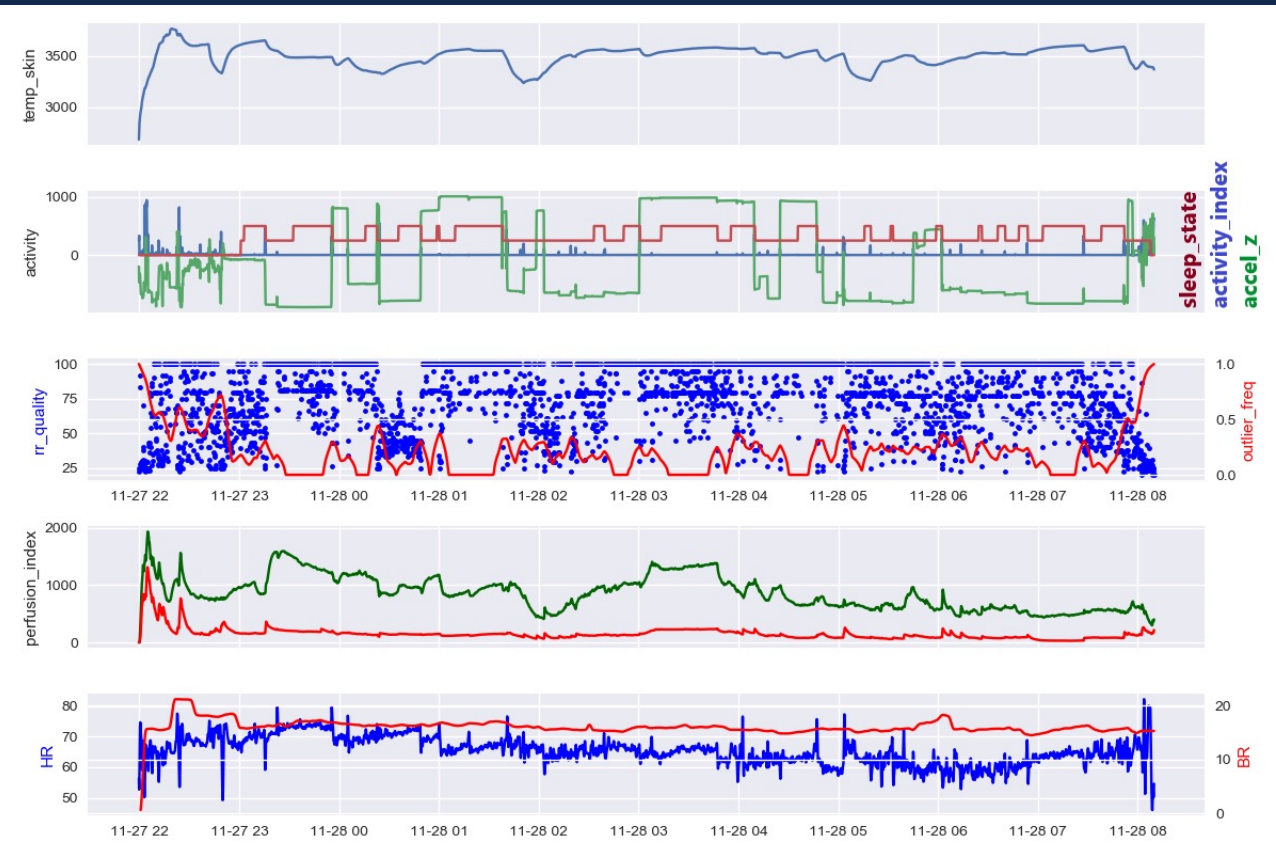


Parse, split, filter
Extract nightly features
Assess quality

	EXISTING
Production code	Python module for data processing
Scaling solution	Spark on AWS
Challenges	- Spark own language - Re-implement solution - Maintain 2 parallel implementations



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	- Spark own language - Re-implement solution - Maintain 2 parallel implementations	- Additional implementation of infrastructure to run parallelisation



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

- “Glue code”
- Re-implementation of code

Solutions

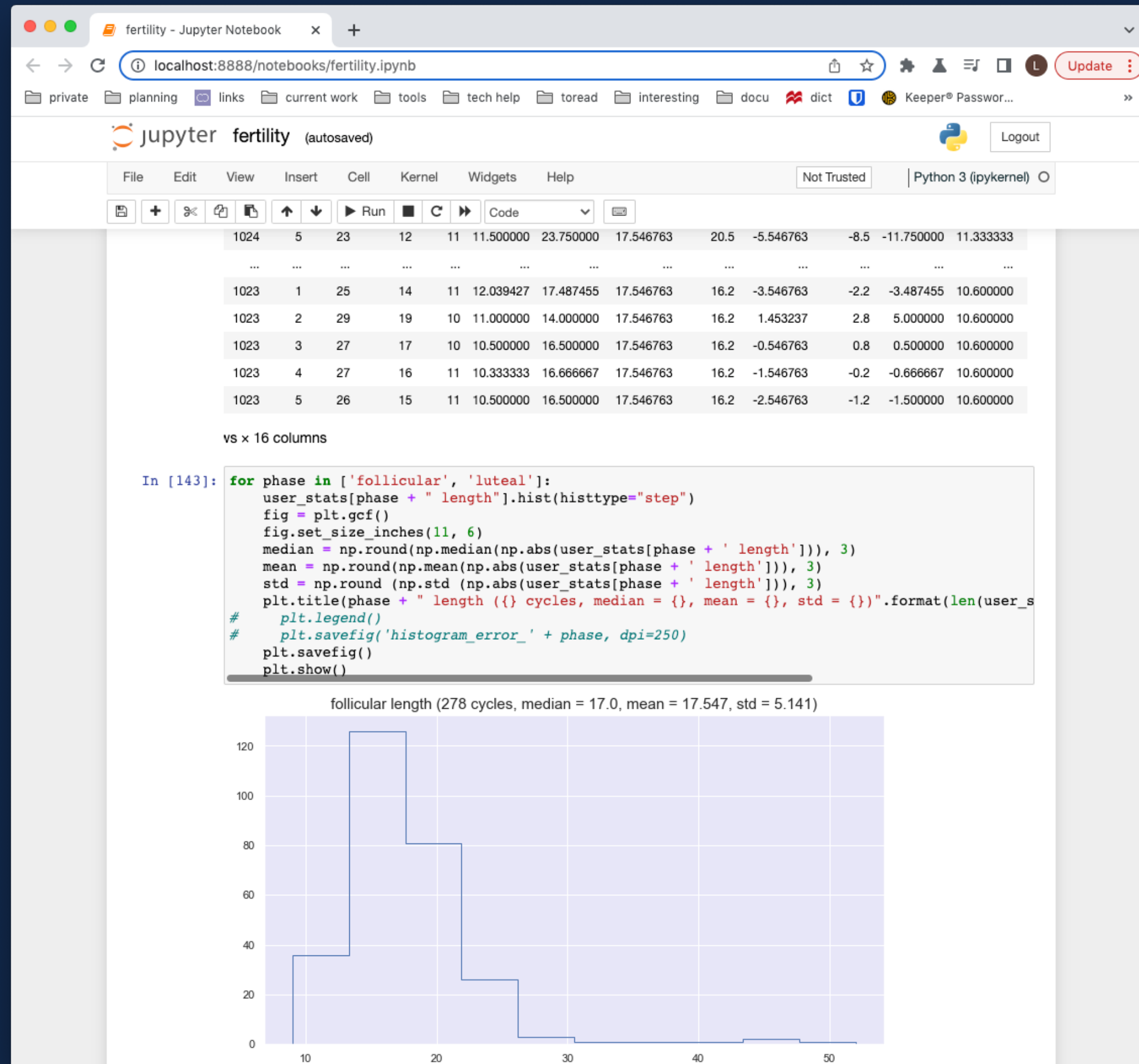
Just a question of writing good code, right?

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)

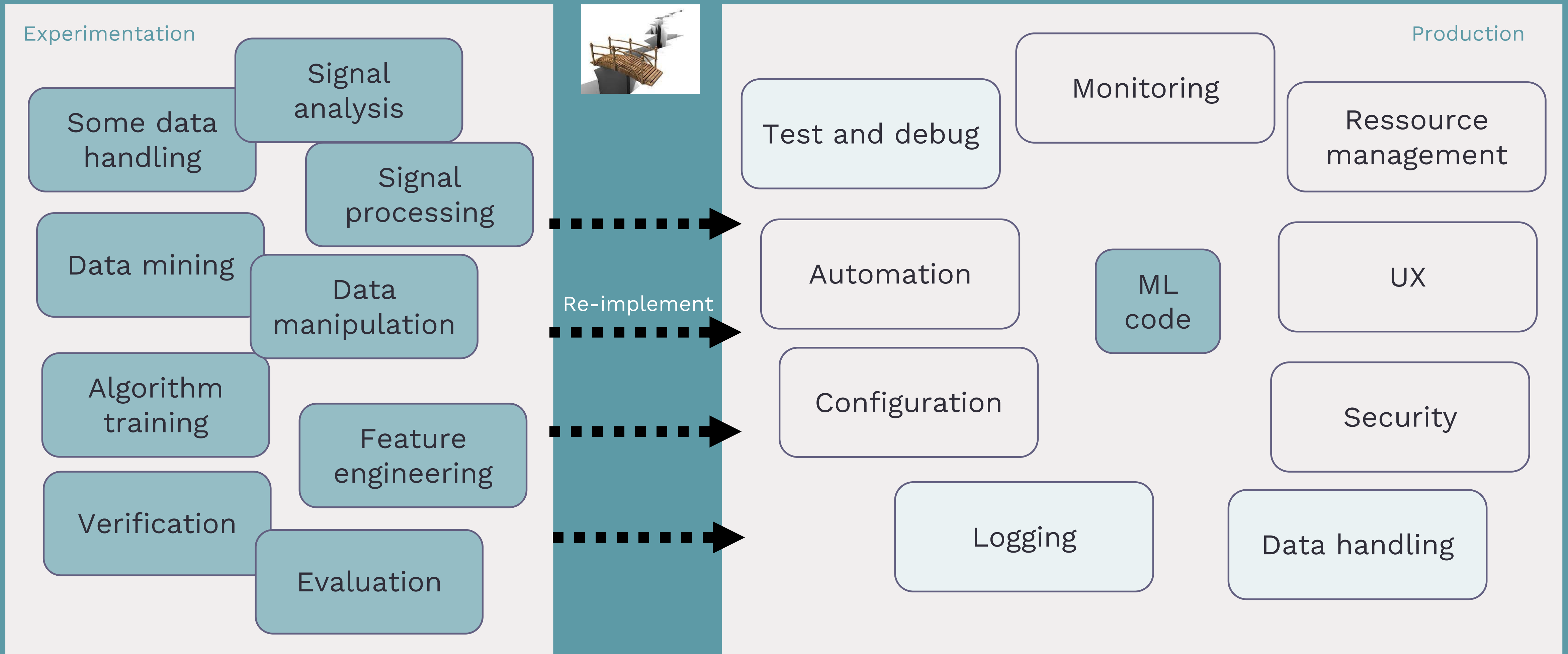


Transitioning from research code to production code

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)





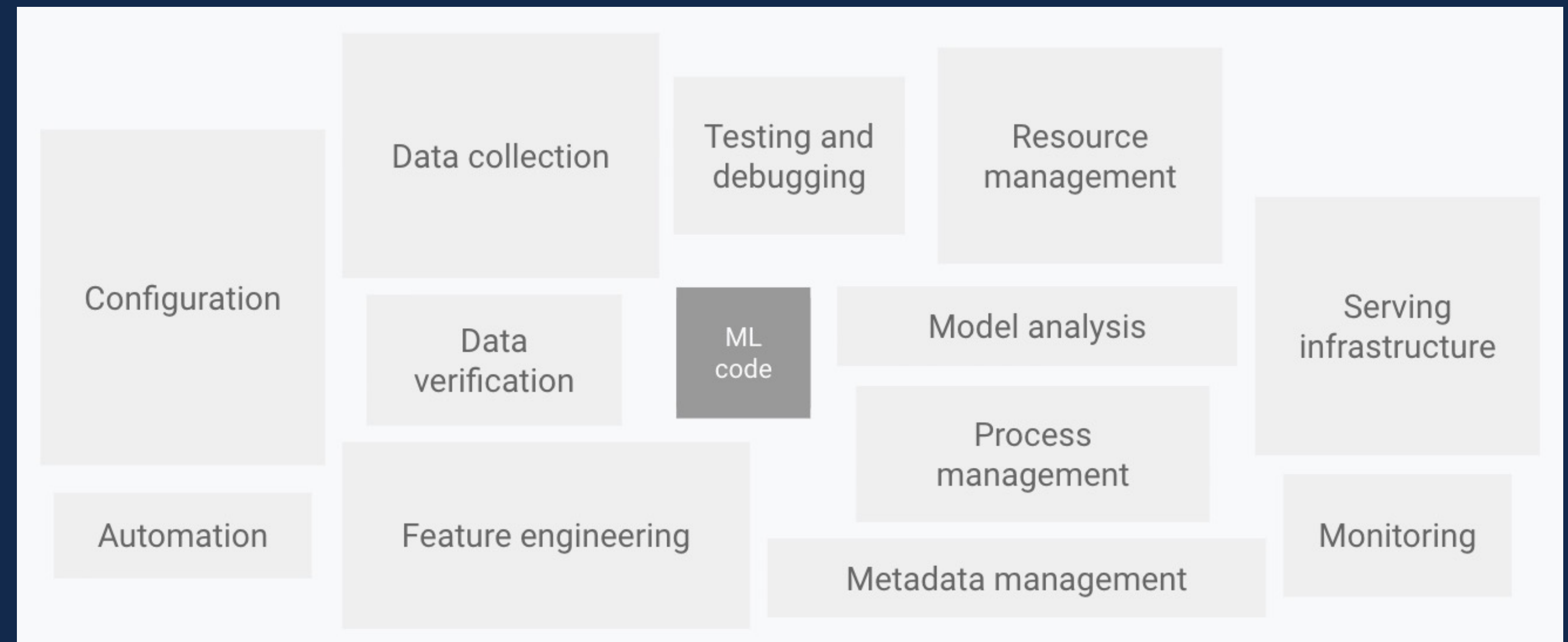
Required infrastructure surrounding lifecycle of ML code

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

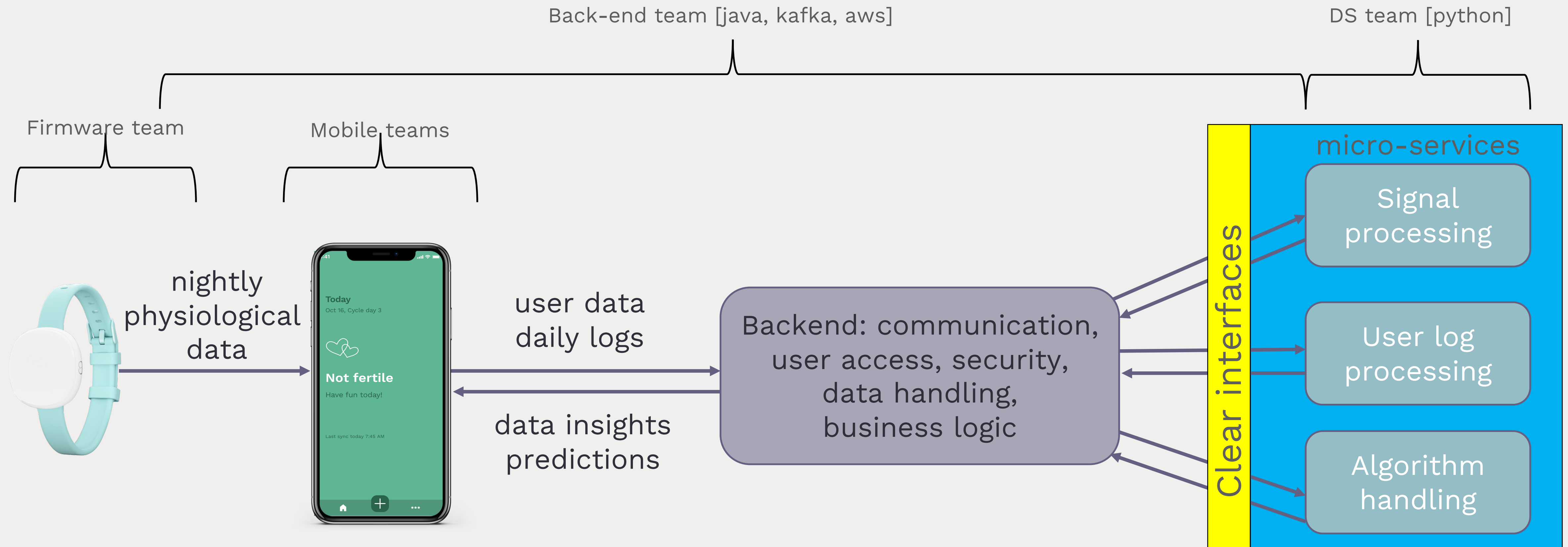
Depends...

SW glue: yes!

ML glue: no!

- Parallel implementations
- Inconsistency problems
- Time to market

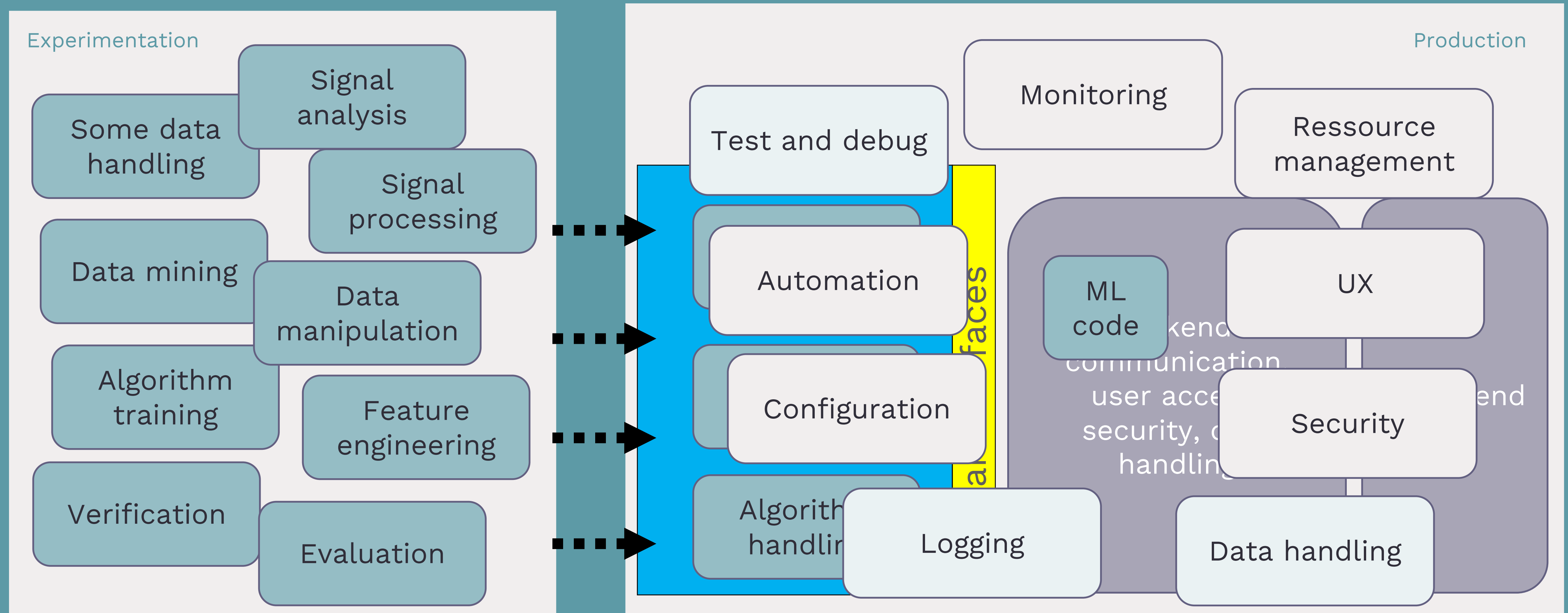
High level architecture



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



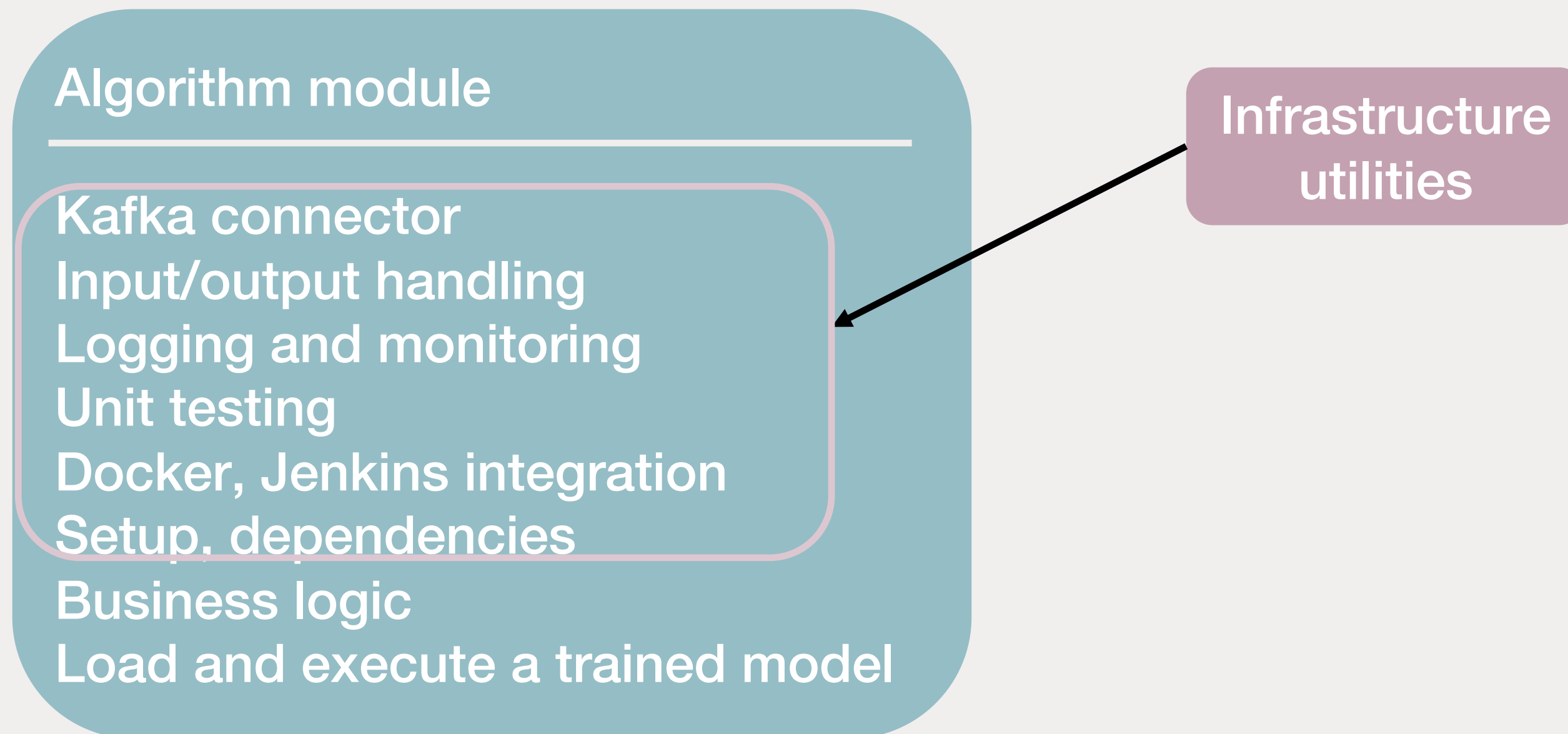
ML modules as microservices



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

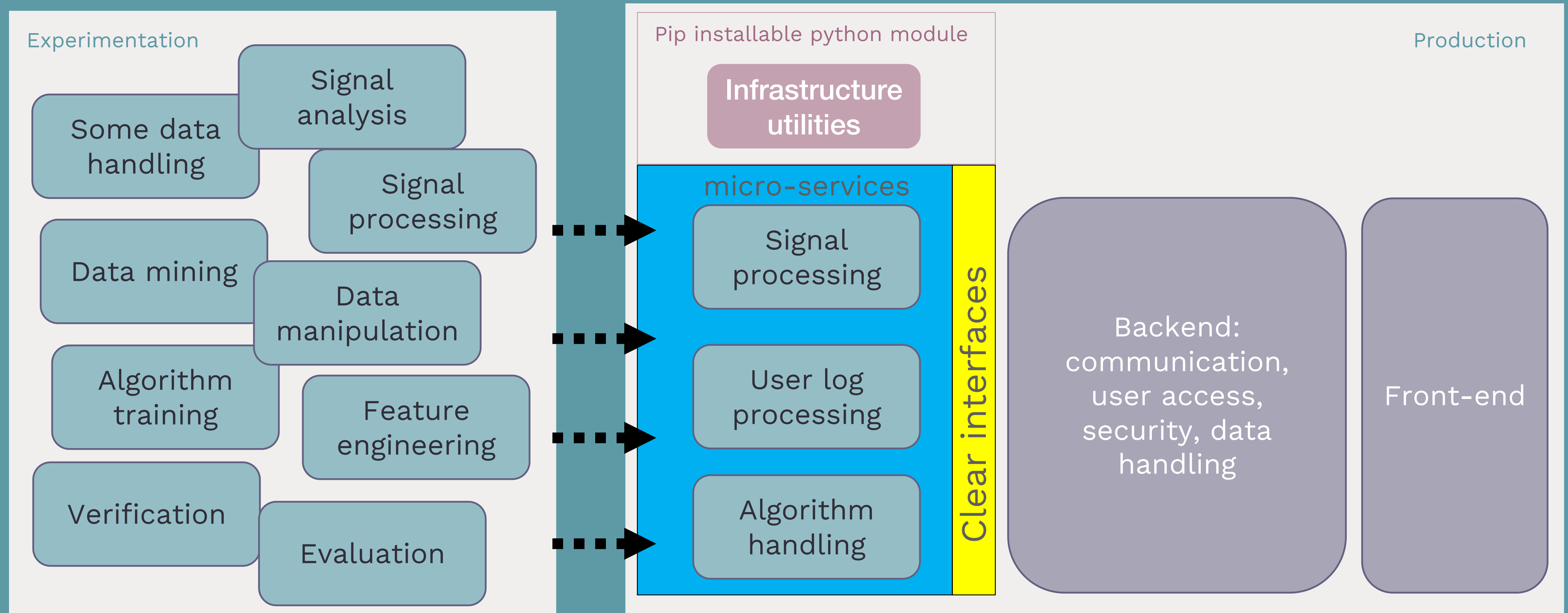


Modularise the DS modules





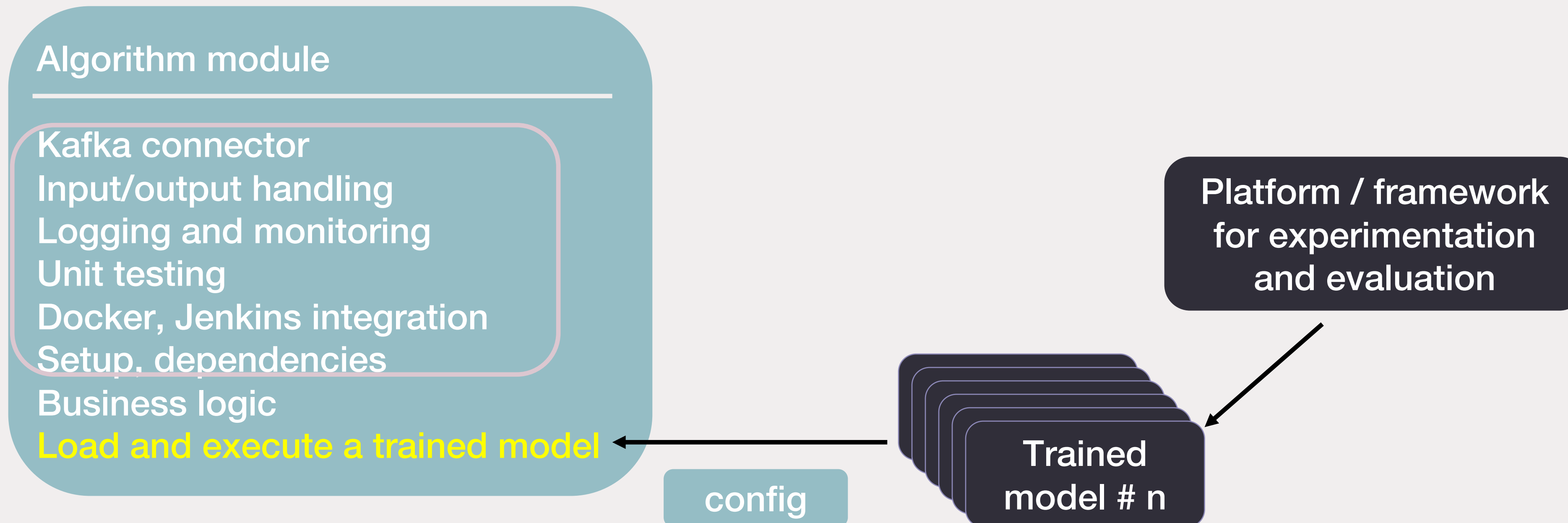
Microservices and modularising the common code



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



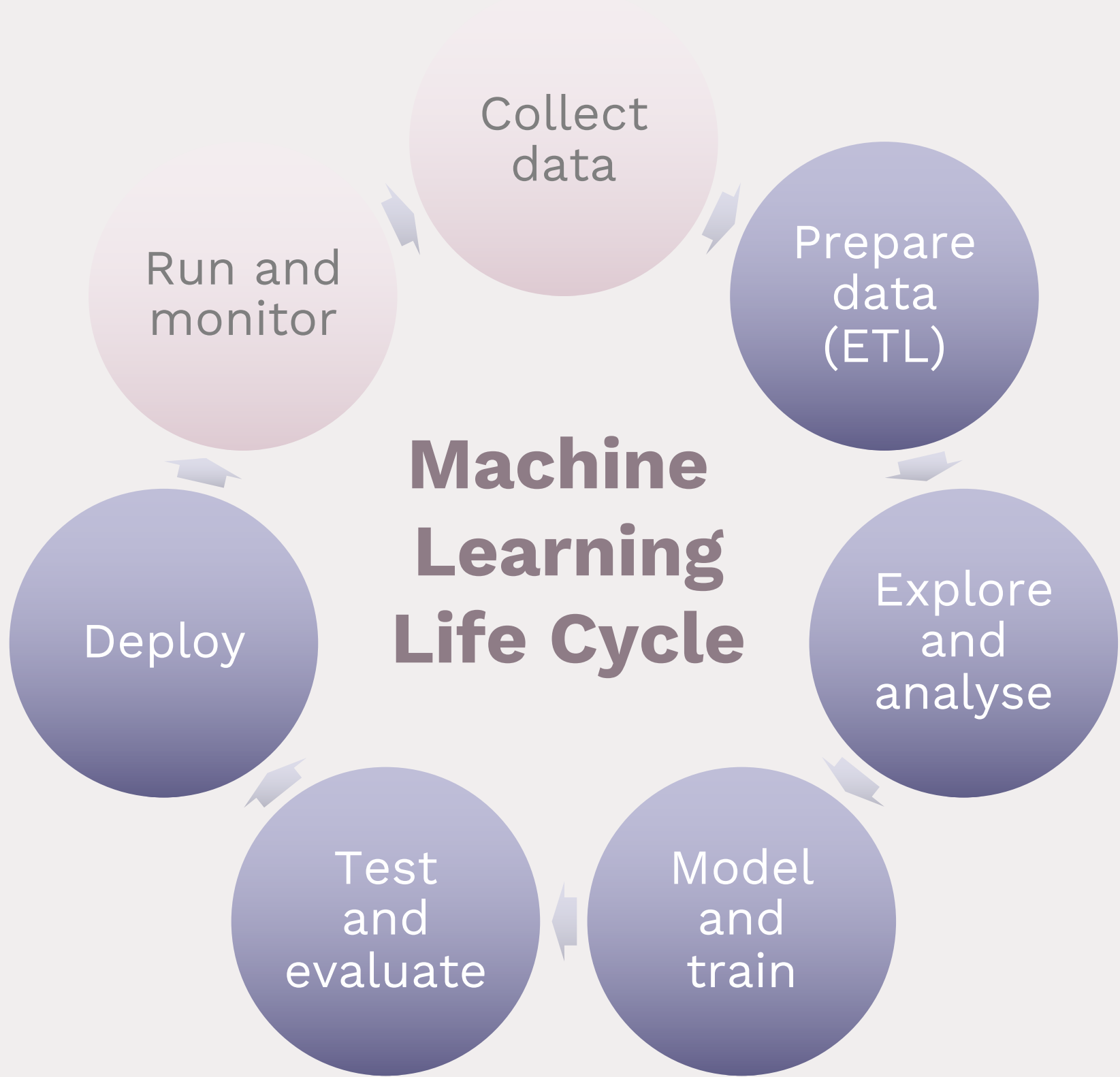
Inject trained models



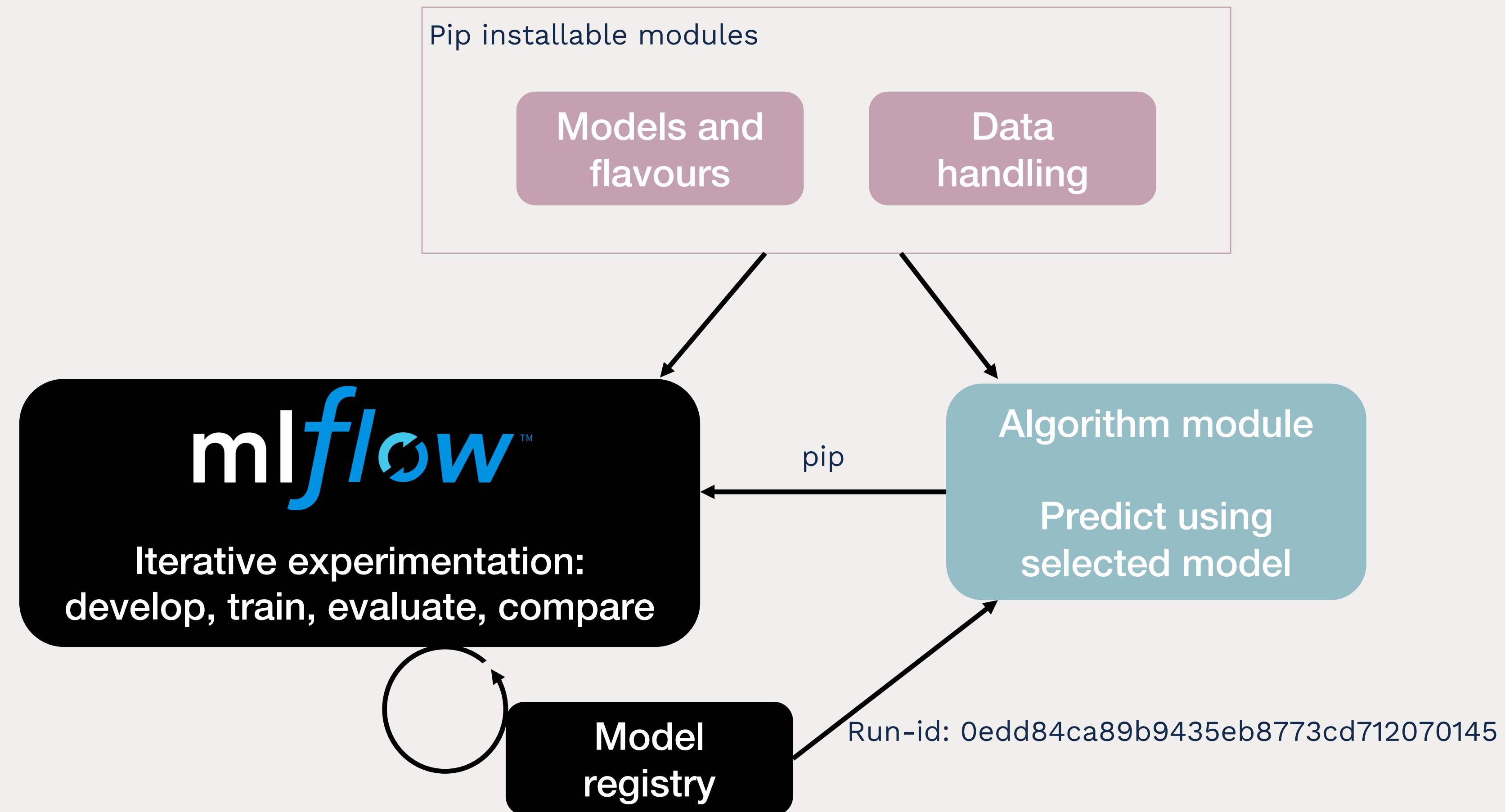


Most popular ml-ops tools for managing ML life cycle

	
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



mlflow Experiments Models

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

🔍 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_pipe	117df2	lowess-gri.../4	-2.597	0.35	(			0.366	-2.629	0.175	se...	4db38753...
☐	🟢 3 hours agc	4.8min	-	linda	train_pipe	b4dfb3	lowess-gri.../3	-	0.361	-			0.359	-2.652	0.185	se...	4db38753...
☐	🟢 3 hours agc	4.4min	-	linda	train_pipe	b4dfb3	lowess-gri.../2	-	0.344	-			0.364	-2.629	0.243	se...	4db38753...
☐	🟢 4 hours agc	4.5min	-	linda	train_pipe	b4dfb3	lowess-gri.../1	-	0.362	-			0.358	-2.657	0.186	se...	4db38753...
☐	🟢 4 hours agc	21.7s	-	linda	train_pipe	b4dfb3	lowess-gri.../22	-	0.333	-			0.38	-3.167	0.194	se...	08629a84...
☐	🟢 4 hours agc	11.6s	-	linda	train_pipe	1b6957	lowess-gri.../20	-	0.556	-			0.257	-2.222	0.181	se...	08629a84...
☐	🟢 5 hours agc	8.8s	-	linda	train_pipe	1b6957	lowess-gri.../19	-	0.444	-			0.291	-2.444	0.194	se...	08629a84...
☐	🟢 5 hours agc	8.1s	-	linda	train_pipe	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

mlflow Experiments Models

train_pipe_0613 > Run fe838e59bfc149daa4a83960fc0c465

Run fe838e59bfc149daa4a83960fc0c465

Date: 2022-06-13 21:41:43 Source: train_pipeline.py Git Commit: 117df2e2bbd5c0

User: linda Duration: 6.1min Status: FINISHED

Lifecycle Stage: active

Description [Edit](#)

Parameters (3)

Metrics (8)

Tags

Artifacts

closing-model-pipeline

Full Path: /minusr/4/fe838e59bfc149daa4a83960fc0c465/artifacts/ov+1_distance_distribution.png

Size: 18.28KB

ov+1_distance_distribution.png

raw_scores.csv

raw_scores.txt

mlflow Experiments Models

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

The image shows a software interface with two main panels. The left panel is a file explorer for a project named 'closing_algo'. It shows a directory structure with folders like 'data', 'datahandler', 'evaluate_algoworker', and 'mlruns'. Under 'mlruns', there is a folder '0' containing several subfolders. One subfolder, '0edd84ca89b9435eb8773cd712070145', is expanded to show its contents: 'artifacts' (containing 'closing-model'), 'conda.yaml', 'MLmodel' (highlighted in blue), 'model.pkl', and 'requirements.txt'. The right panel is a code editor showing a YAML file 'mlruns/.../conda.yaml'. The file contains configuration for an MLflow artifact, including the artifact path, flavors (python function, environment, loader module, model path, and python version), sklearn configuration (pickled model, serialization format, and sklearn version), and metadata (model UUID, run ID, and creation time).

```
1 artifact_path: closing-model
2 flavors:
3   python_function:
4     env: conda.yaml
5     loader_module: mlflow.sklearn
6     model_path: model.pkl
7     python_version: 3.7.11
8   sklearn:
9     pickled_model: model.pkl
10    serialization_format: cloudpickle
11    sklearn_version: 1.0.2
12 model_uuid: 2552f3fc84d14f9fbd3c7df4f198adea
13 run_id: 0edd84ca89b9435eb8773cd712070145
14 utc_time_created: '2022-03-22 17:17:04.702535'
```




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

```
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:...
```




Available functionality from *scikit-learn* (*sklearn*)

Standard classes eg:

- RandomForestClassifier
- BaggingClassifier
- GridSearchCV
- StandardScaler
- Pipeline

```
search_grid = close_model.get_parameter_filter(random_grid=True, **extra_kwargs)
grid_search_classifier = RandomizedSearchCV(
    close_model, search_grid, scoring=model_scorer, n_iter=int(n_iter)
)

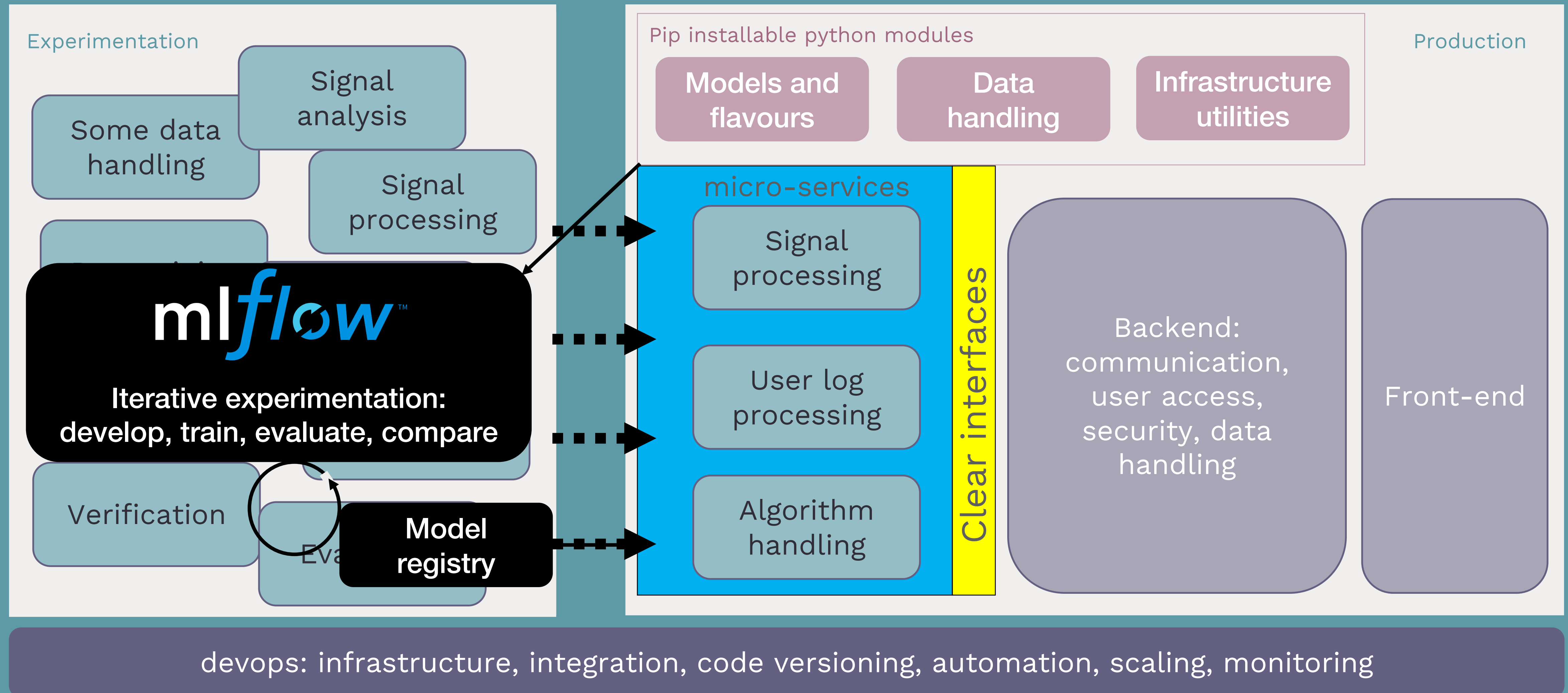
pipeline = Pipeline(
    [
        ("smoother", LowessSignalSmoother(lowess_look_back_period, lowess_frac)),
        ("grid_search_classifier", grid_search_classifier),
    ]
)

pipeline.fit(cycle_lists[TRAIN_SPLIT_NAME])
mlflow.log_metrics(grid_search_classifier.best_params_)
df = pd.DataFrame(grid_search_classifier.cv_results_)
store_artifact_local_dump(
    artifact_name="cv_results.csv", dump_function=df.to_csv, index=False
)

score = pipeline.score(cycle_lists[TRAIN_SPLIT_NAME])
```




Mlflow for experimentation and model serving



Summary

And challenges

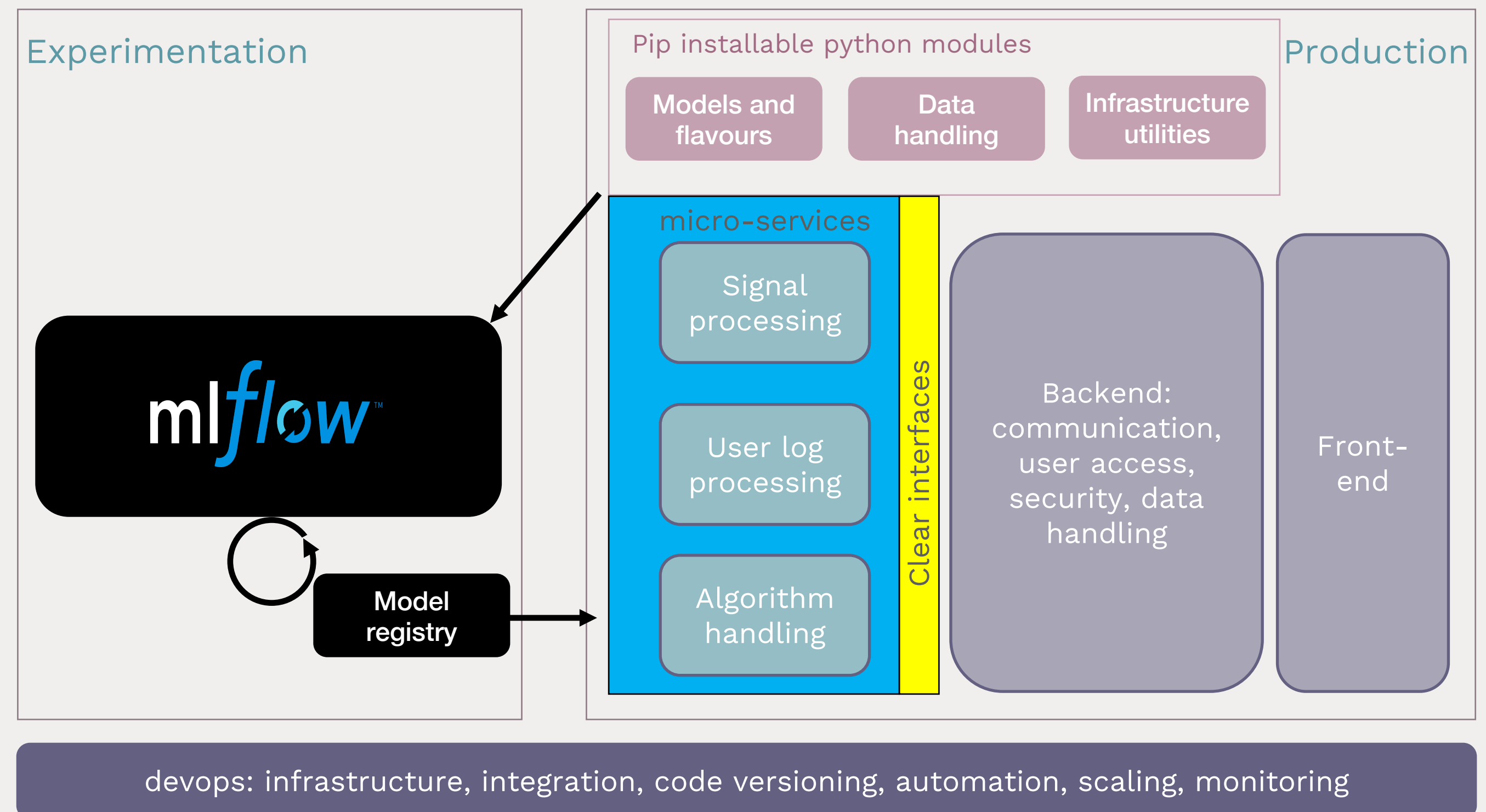


Solution at ava

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

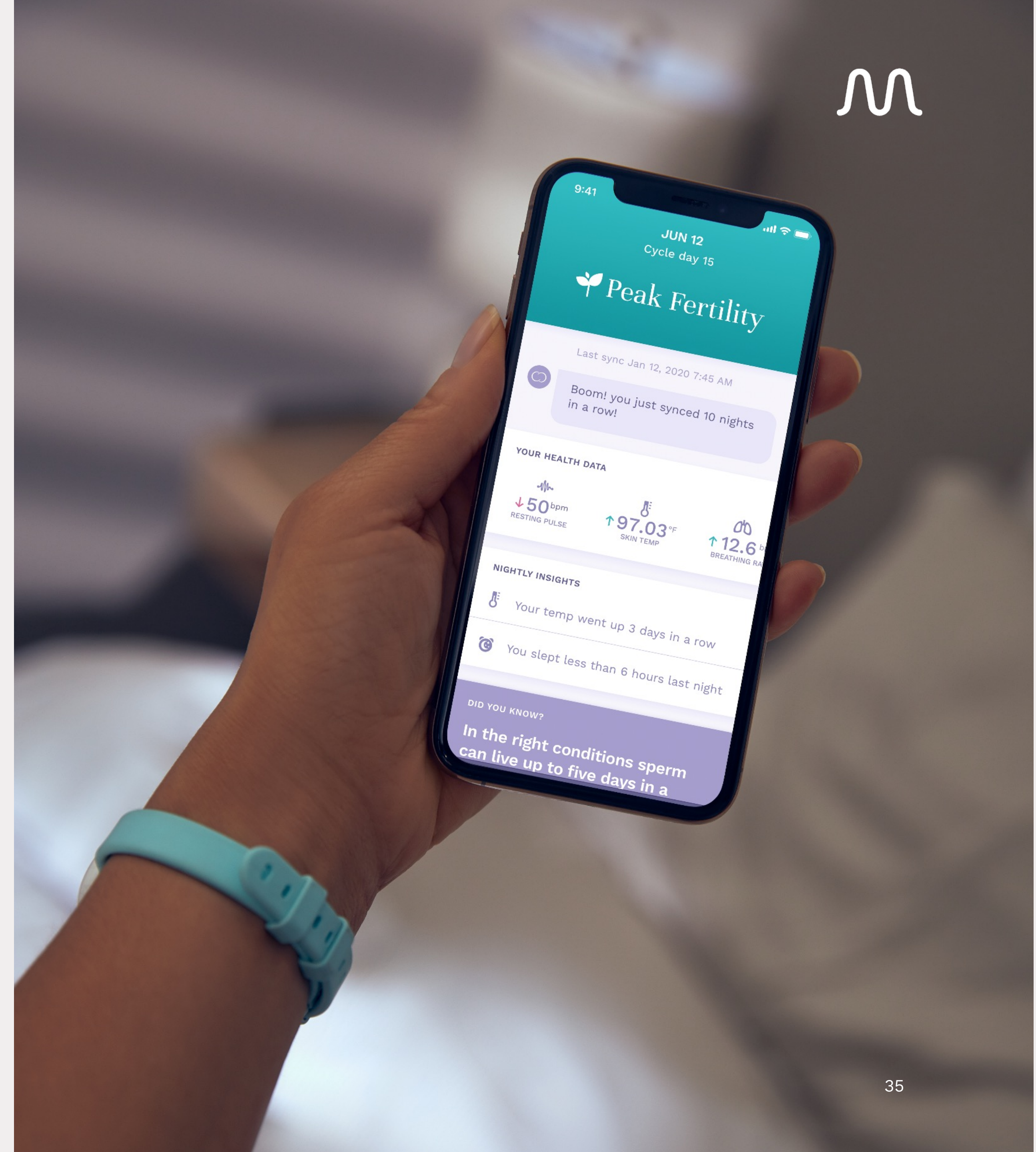
It is out of comfort zone for most researchers

- 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?