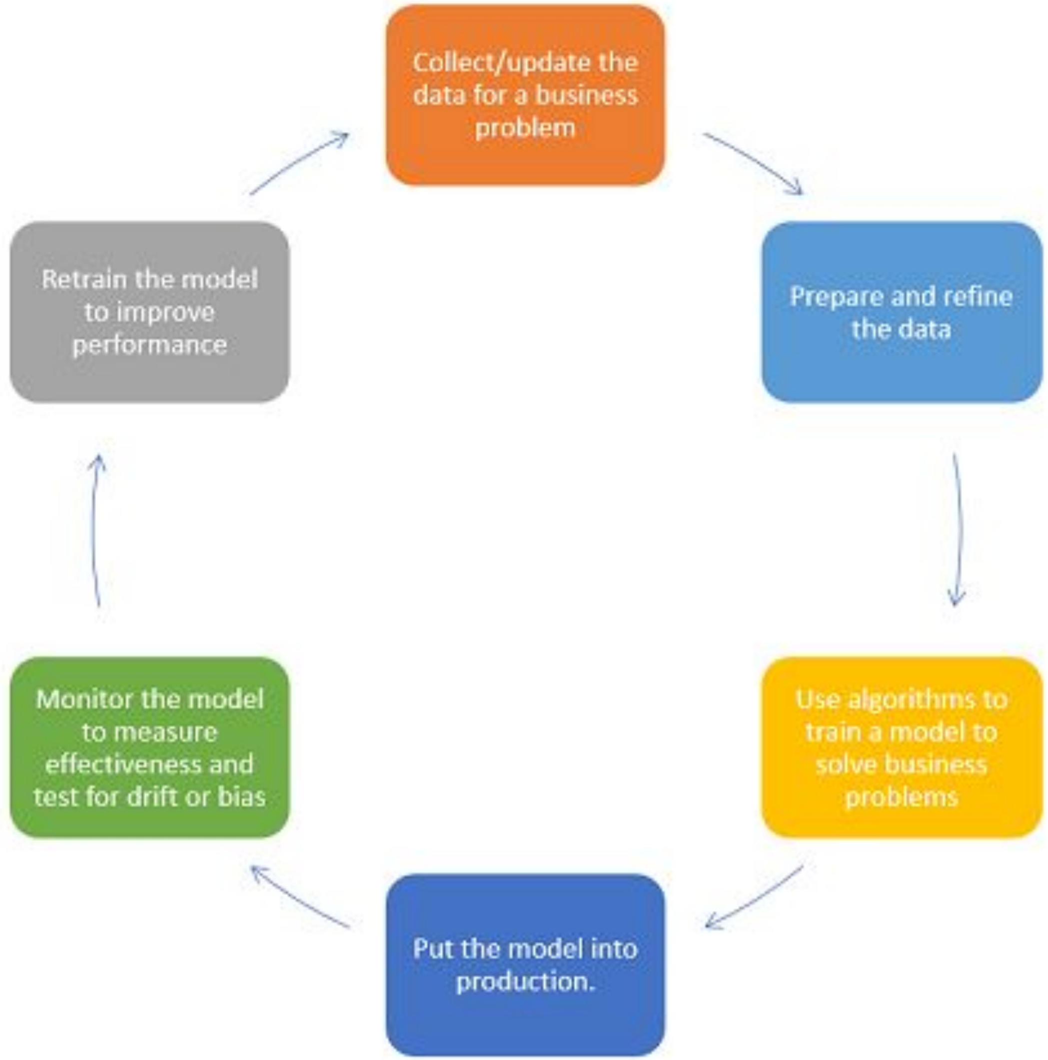
ML Lifecycle

Why understand ML Lifecycle and challenges encountered during the lifecvcle?

Most Enterprises Are Failing to Scale AI

In 2018, a team at Gartner research asked large enterprises the amount of AI adoption that they expect to deploy in the next 12 months. The enterprises said that they were planning to deploy AI in 23% of their systems. Then, in 2019, they went back to check how many of these projects actually deployed. The team found out that only 5% of the AI adoptions that enterprises wanted to deploy were actually deployed.



4 Phases to ML Lifecycle

Phase 1 is Project Planning and Project Setup: At this phase, we want to decide the problem to work on, determine the requirements and goals, as well as figure out how to allocate resources properly.

Phase 2 is Data Collection and Data Labeling: At this phase, we want to collect training data (images, text, tabular, etc.) and potentially annotate them with ground truth, depending on the specific sources where they come from.

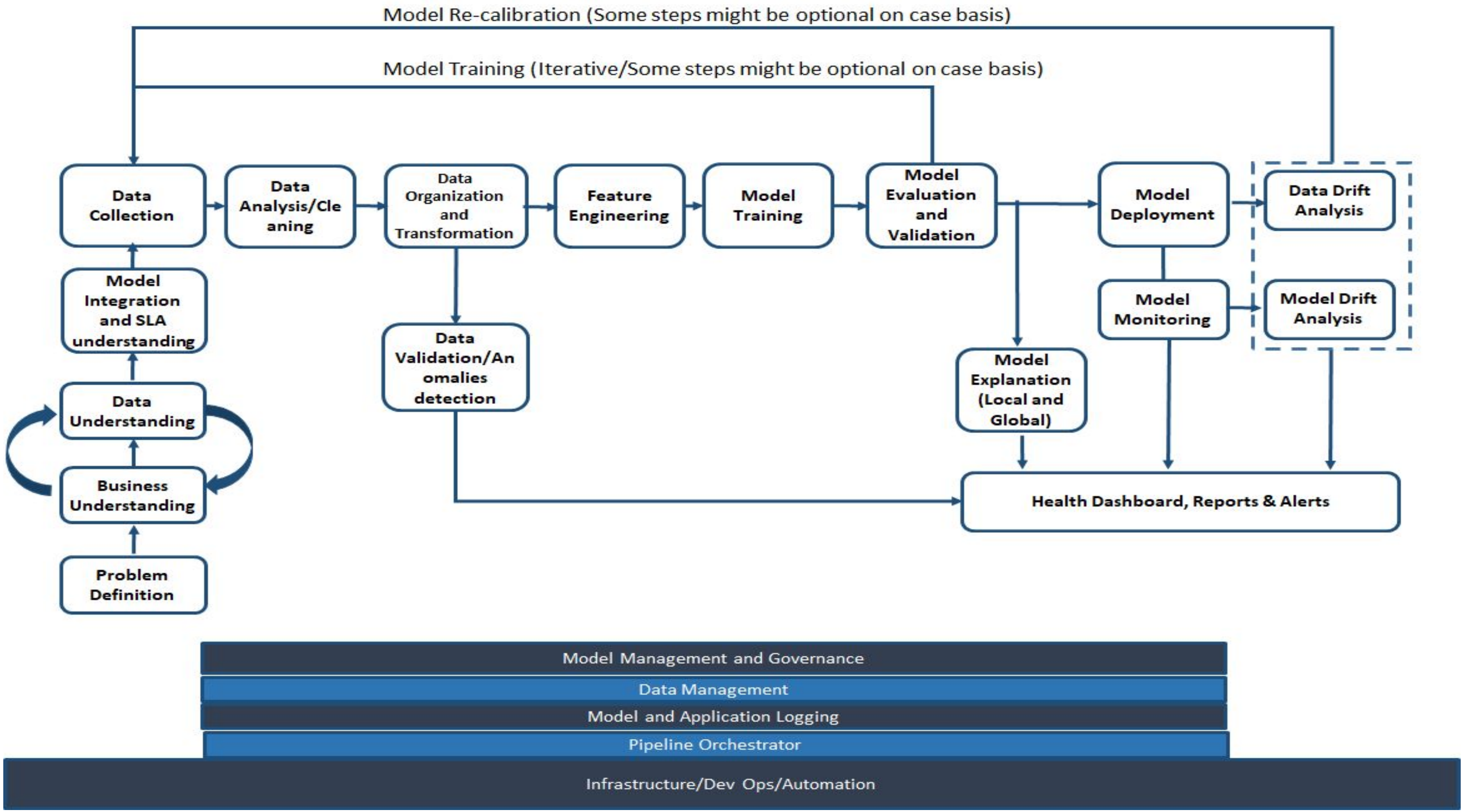
Phase 3 is Model Training and Model Debugging: At this phase, we want to implement baseline models quickly, find and reproduce state-of-the-art methods for the problem domain, debug our implementation, and improve the model performance for specific tasks.

Phase 4 is Model Deployment and Model Testing: At this phase, we want to pilot the model in a constrained environment, write tests to prevent regressions, and roll the model into production.

Ref: https://fall2019.fullstackdeeplearning.com/course-content/setting-up-machine-learning-projects/lifecycle

Discussion: How does the real life ML lifecycle looks like?

Real world ML Lifecycle



Challenges with ML during development

➔ Development, training and deployment environment can be different

➔ Tools, libraries and dependencies can complicate deployment

➔ Tracking and analyzing experiment can become tedious to handle

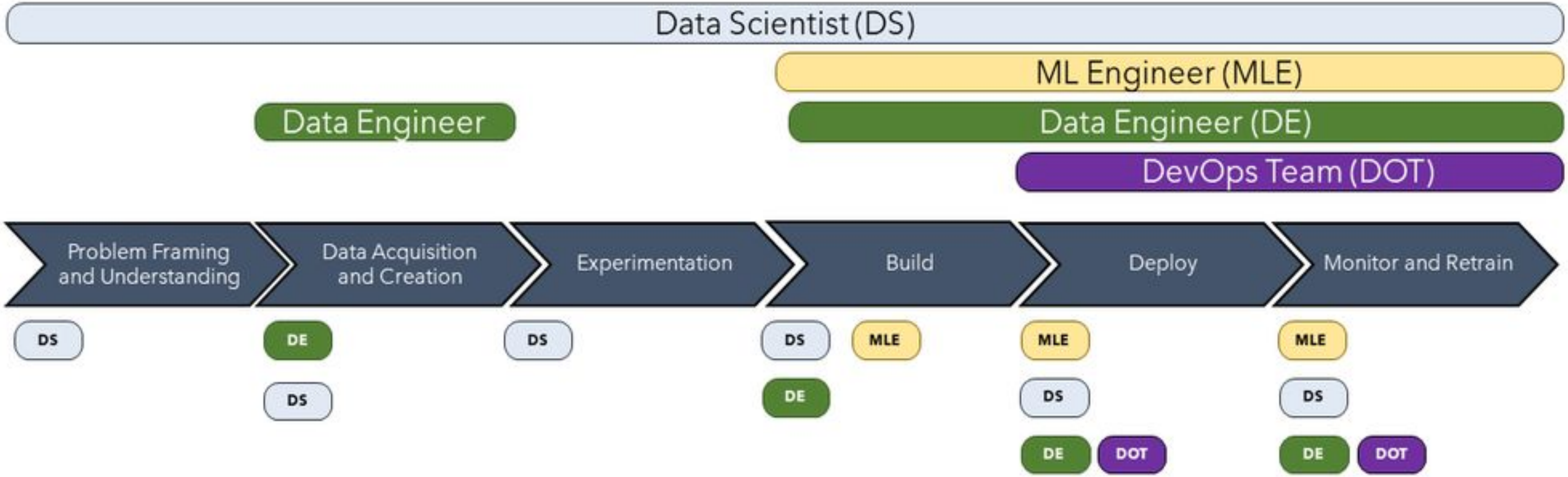
➔ Difficult to reproduce experiment as input data changes

➔ ML Code end up in a spaghetti jungle

Challenges with ML in production

- ➔ Live data is not equal to training data
- ➔ Feature engineering pipeline must match between training and serving infrastructure
- ➔ Seamlessly scale up and scale down deployed model
- ➔ Continuous training and champion challenger model deployment
- ➔ Different technology landscape between development and deployment

Different Roles in ML Lifecycle



References

- [An Introduction to MLOps: AI Engineering]

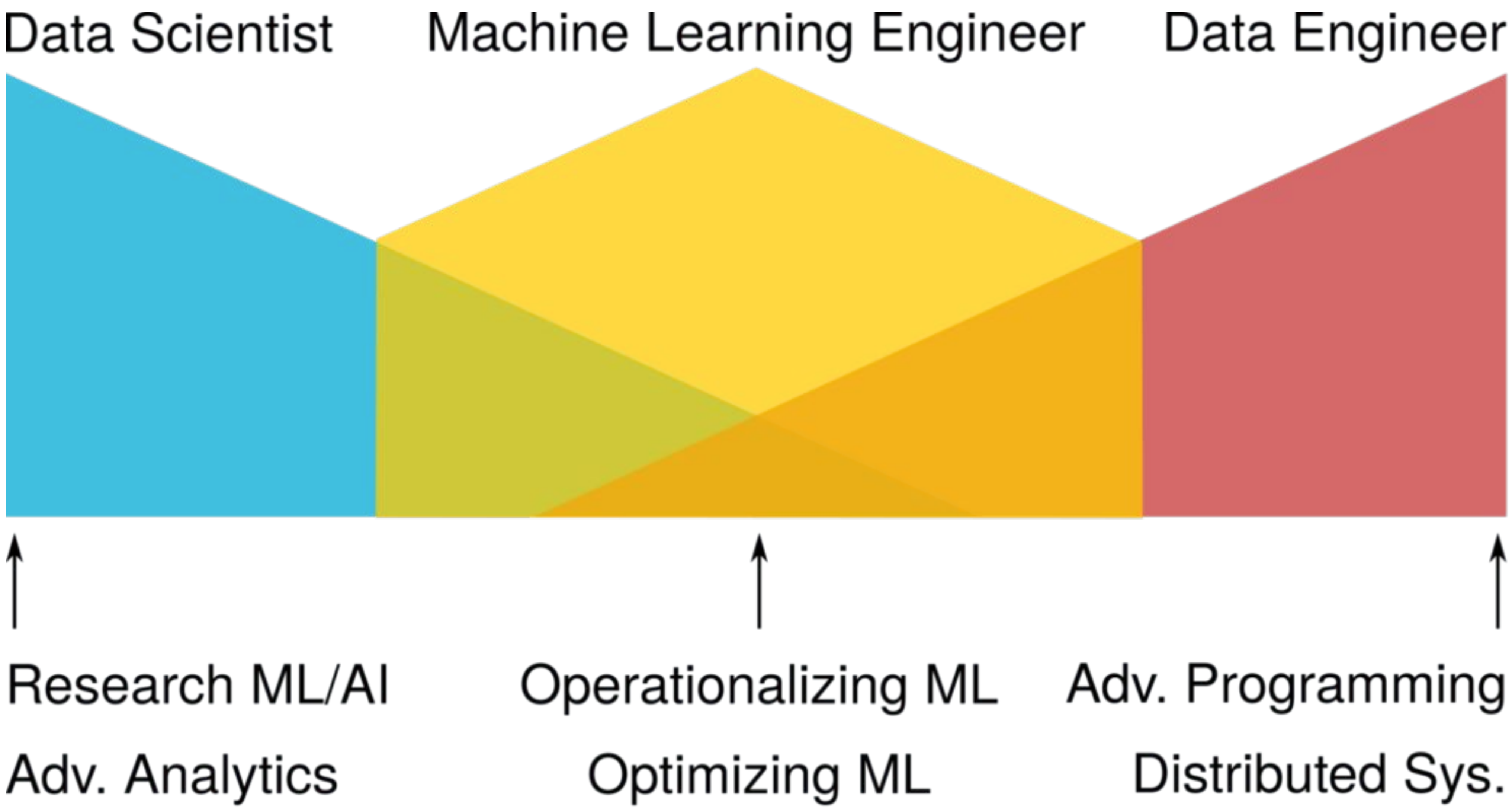
Machine Learning Engineering

Machine Learning Engineering (MLE)

What is machine learning engineering?

Machine learning engineering is the process of using software engineering principles, and analytical and data science knowledge, and combining both of those in order to take an ML model that's created and making it available for use by the product or the consumers.

For example, a YouTube ML engineer might be in charge of developing the next generation YouTube recommendation algorithm and then developing an ML pipeline around it and integrating it into



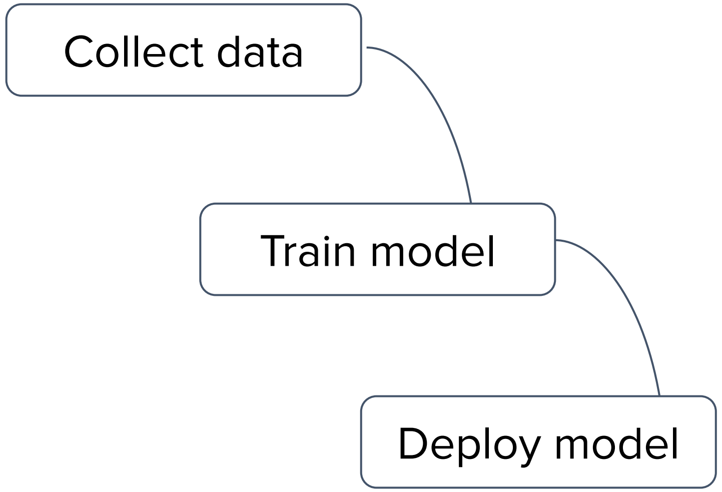
BDI

BIG DATA INSTITUTE

For more information go to http://bigdatainstitute.io

ML in production: expectation

1. Collect data
2. Train model
3. Deploy model

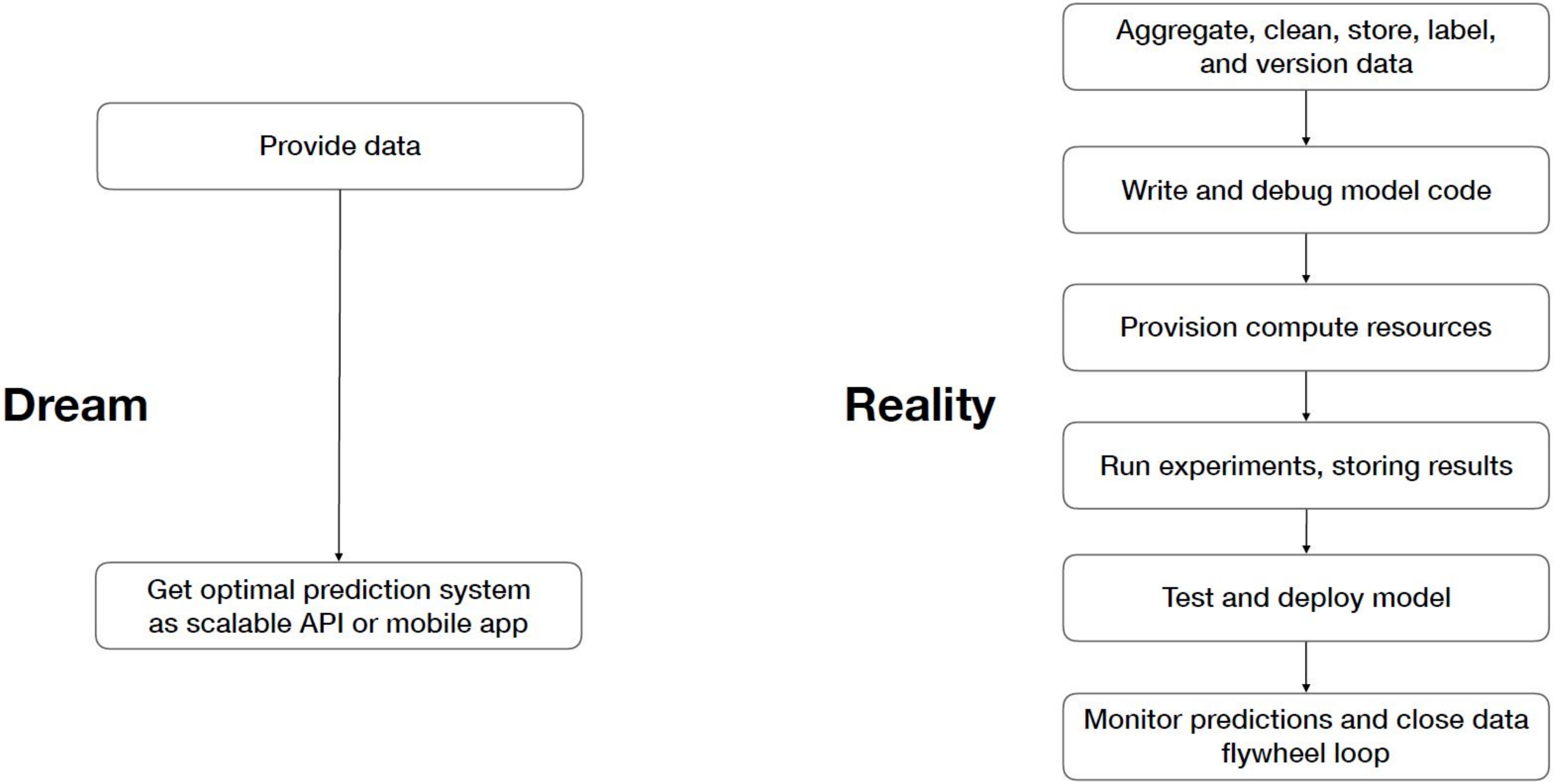


Waterfall model

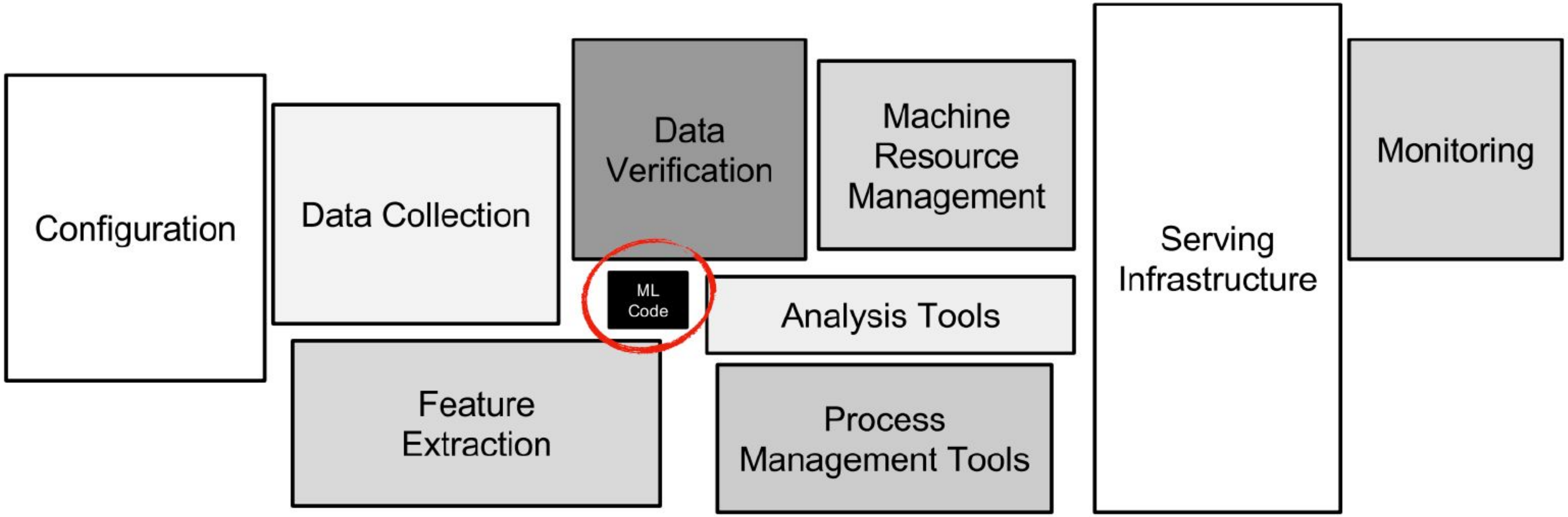
Problems?

- What are the issues in ML in Production?
- Post on https://padlet.com/rajaty6/bpmht0n2n8ls0d6

ML in Production



Machine Learning: Technical Debt

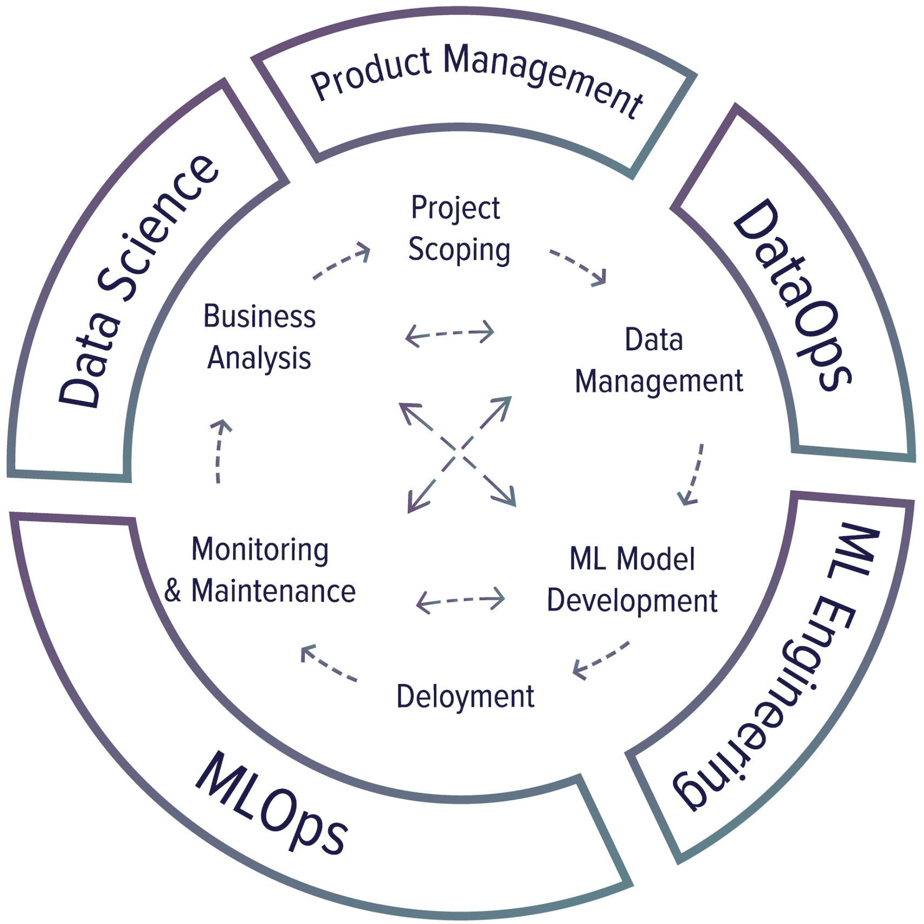


Machine Learning:
The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young

ML in production: reality

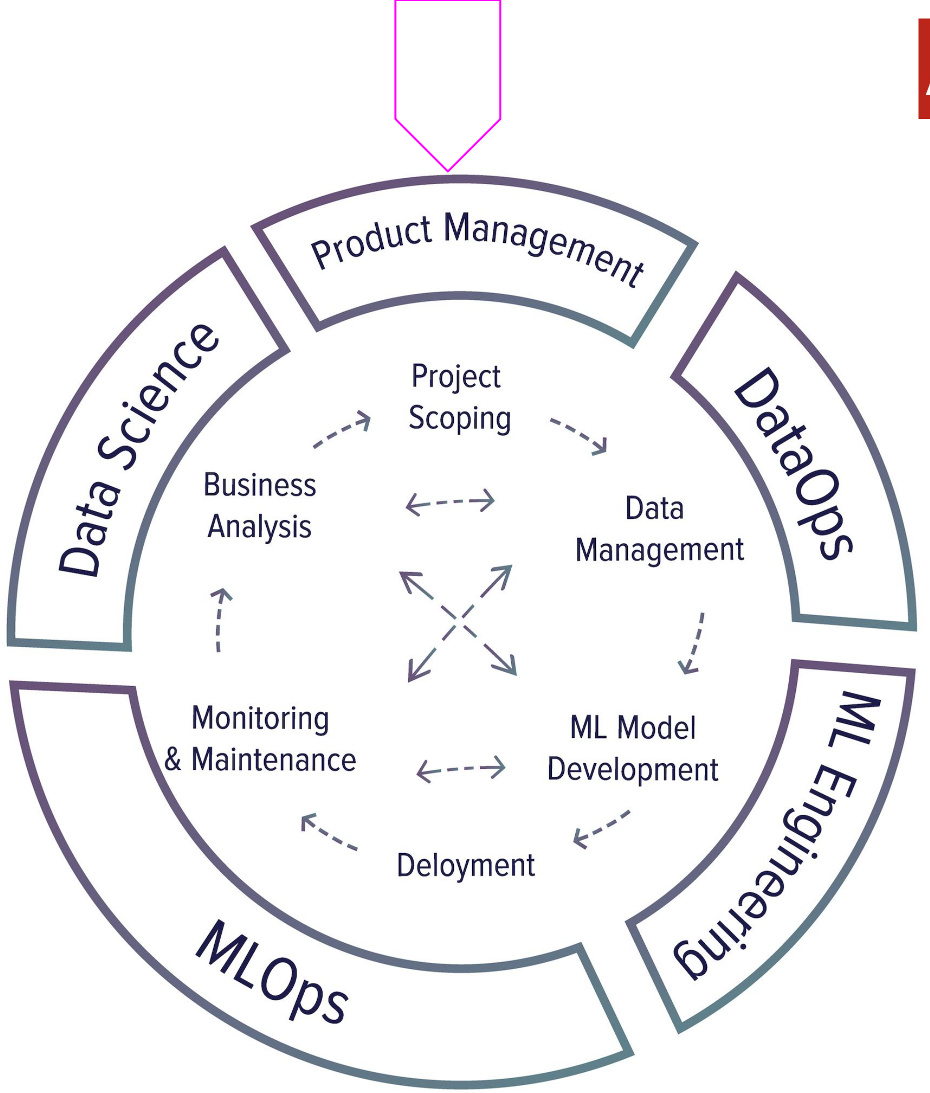
1. Choose a metric to optimize
2. Collect data
3. Train model
4. Realize many labels are wrong -> relabel data
5. Train model
6. Model performs poorly on one class -> collect more data for tha
7. Train model
8. Model performs poorly on most recent data -> collect more rece
9. Train model
10. Deploy model
11. Dream about $$$
12. Wake up at 2am to complaints that model biases against one gr
13. Get more data, train more, do more testing
14. Deploy model
15. Pray
16. Model performs well but revenue decreasing
17. Cry
18. Choose a different metric
19. Start over



Iterative development

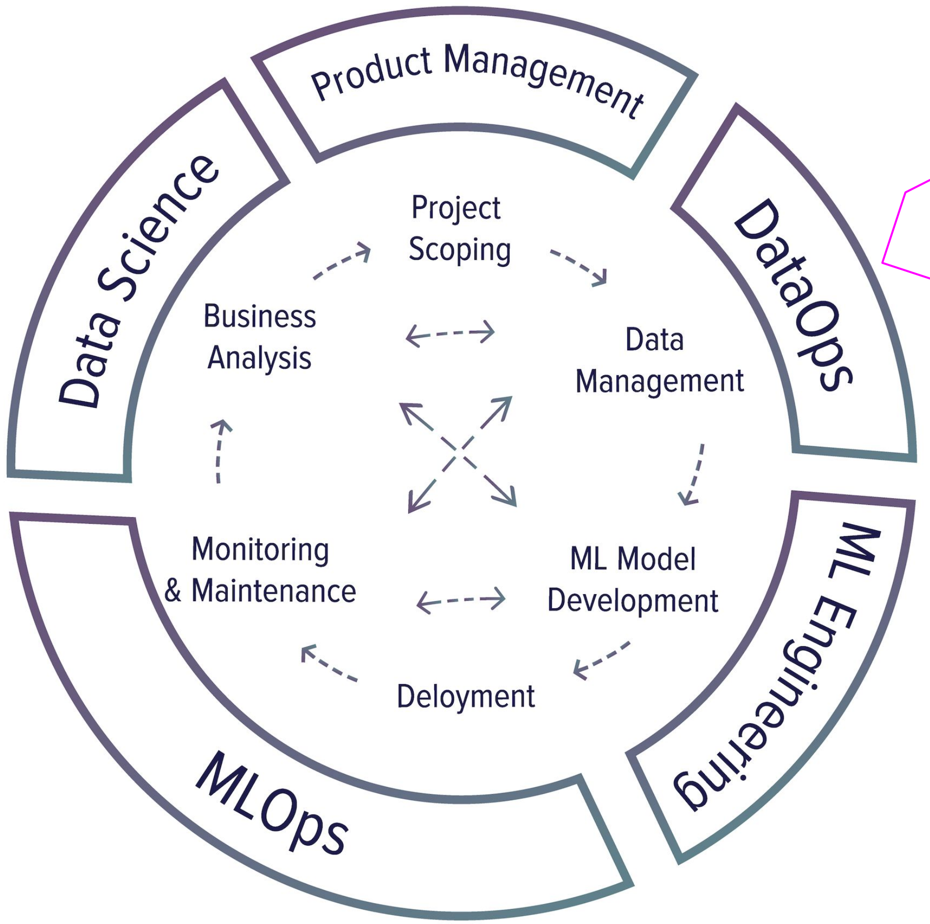
Project scoping

- Goals & objectives
- Constraints
- Evaluation



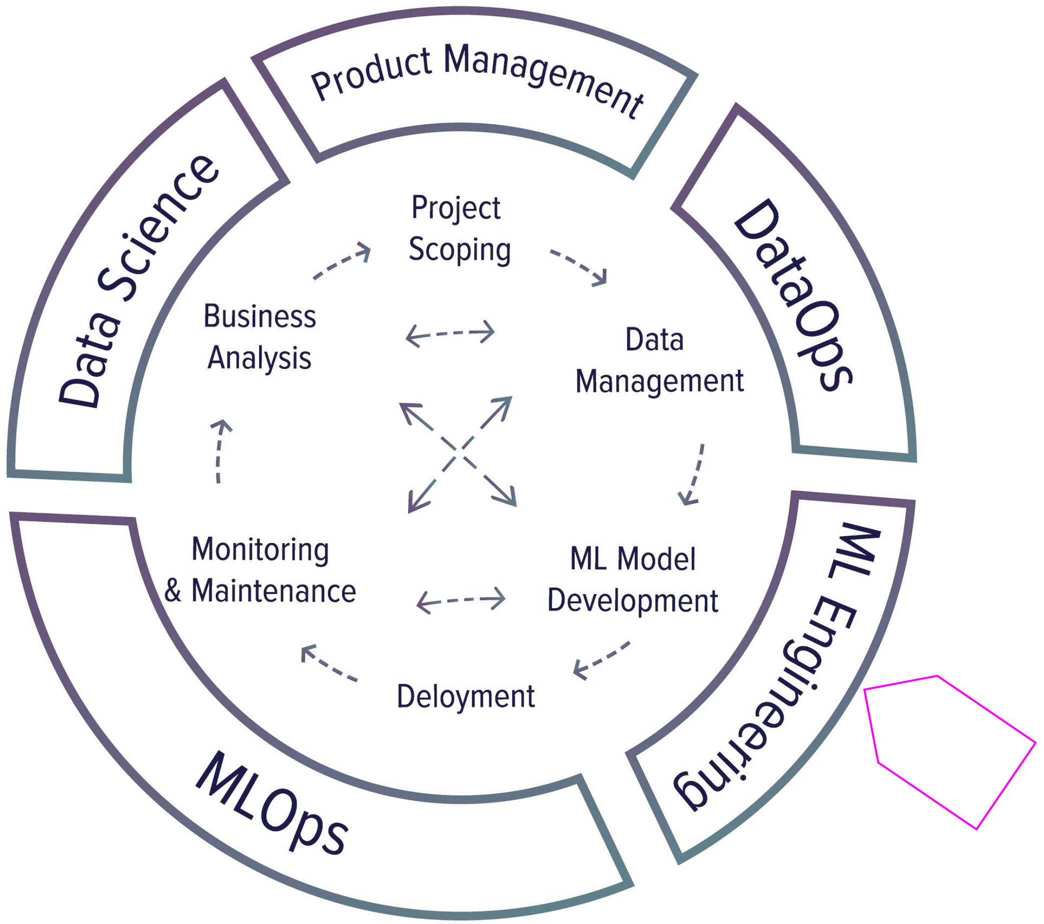
Data management

- Data sources
- Data format
- Processing
- Storage
- Data consumer
- Data controller



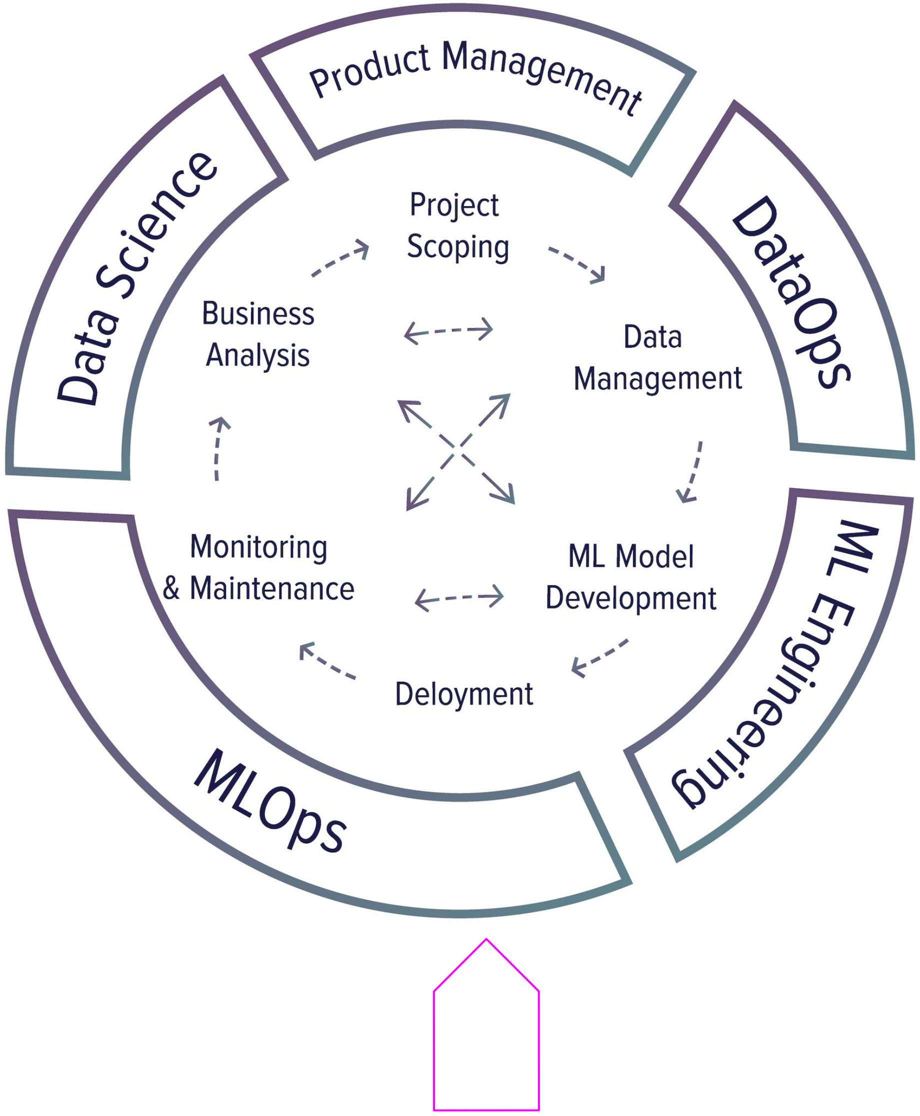
Model development

- Dataset creation
- Feature engineering
- Model training
- Offline model evaluation



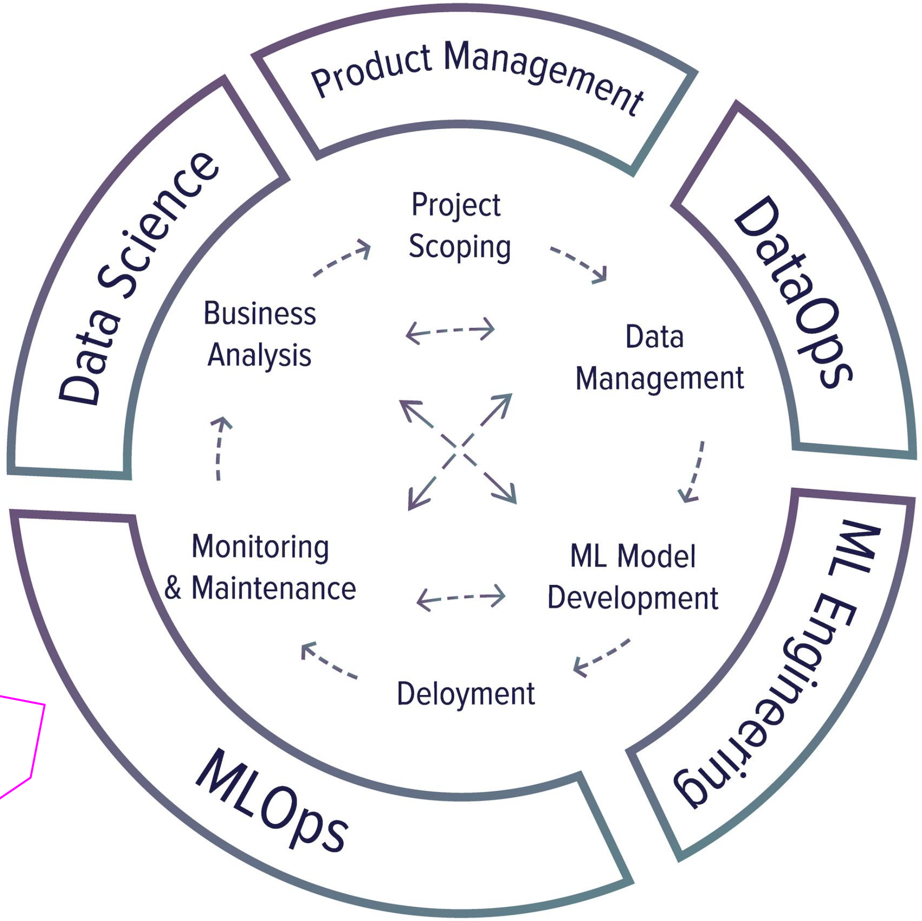
Deployment

- Deploying and serving
- Release strategies
- Online model evaluation



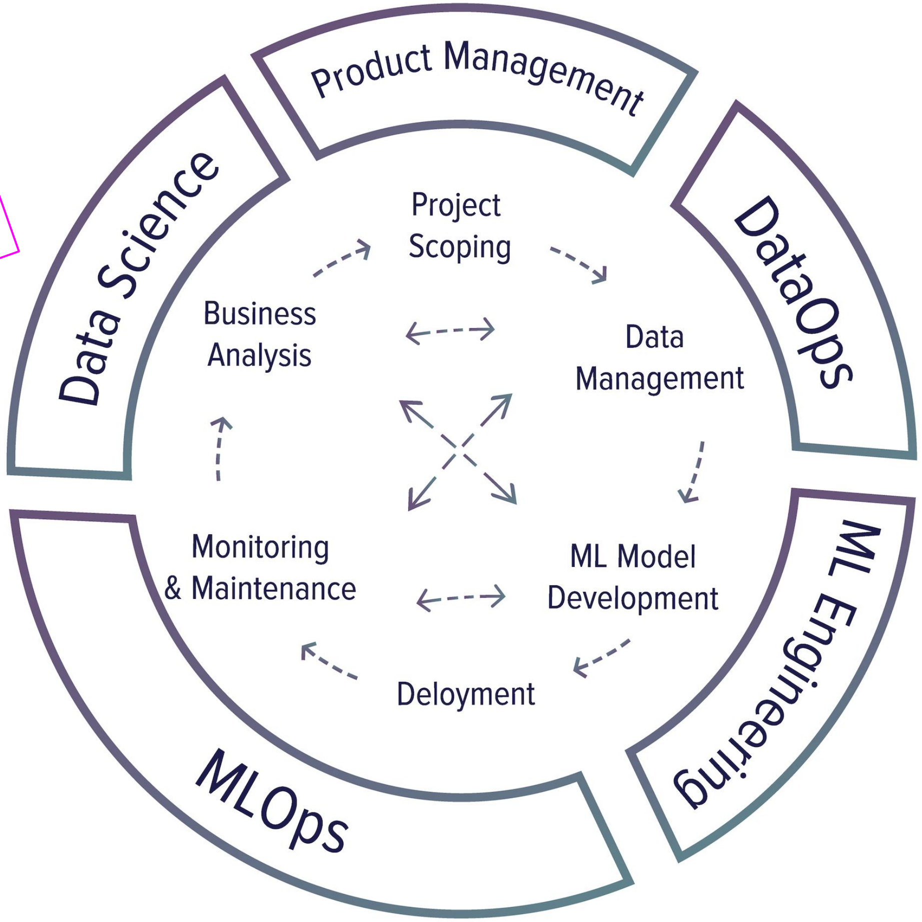
Monitoring & maintenance

- Model performance & data monitoring
- Model retraining
- Model updates



Business analysis

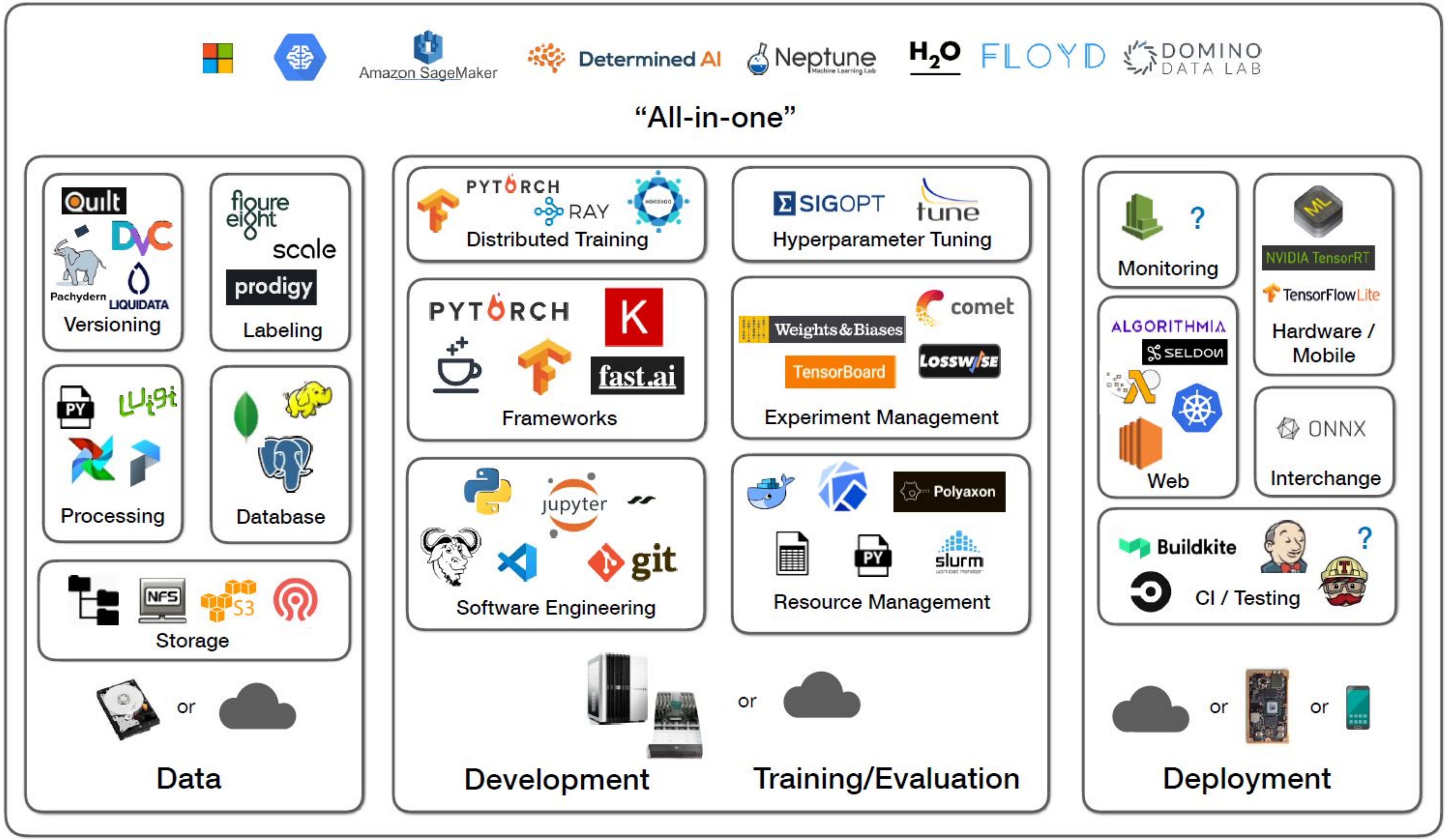
- User experience
- Tying model performance to business performance



Research Vs Production

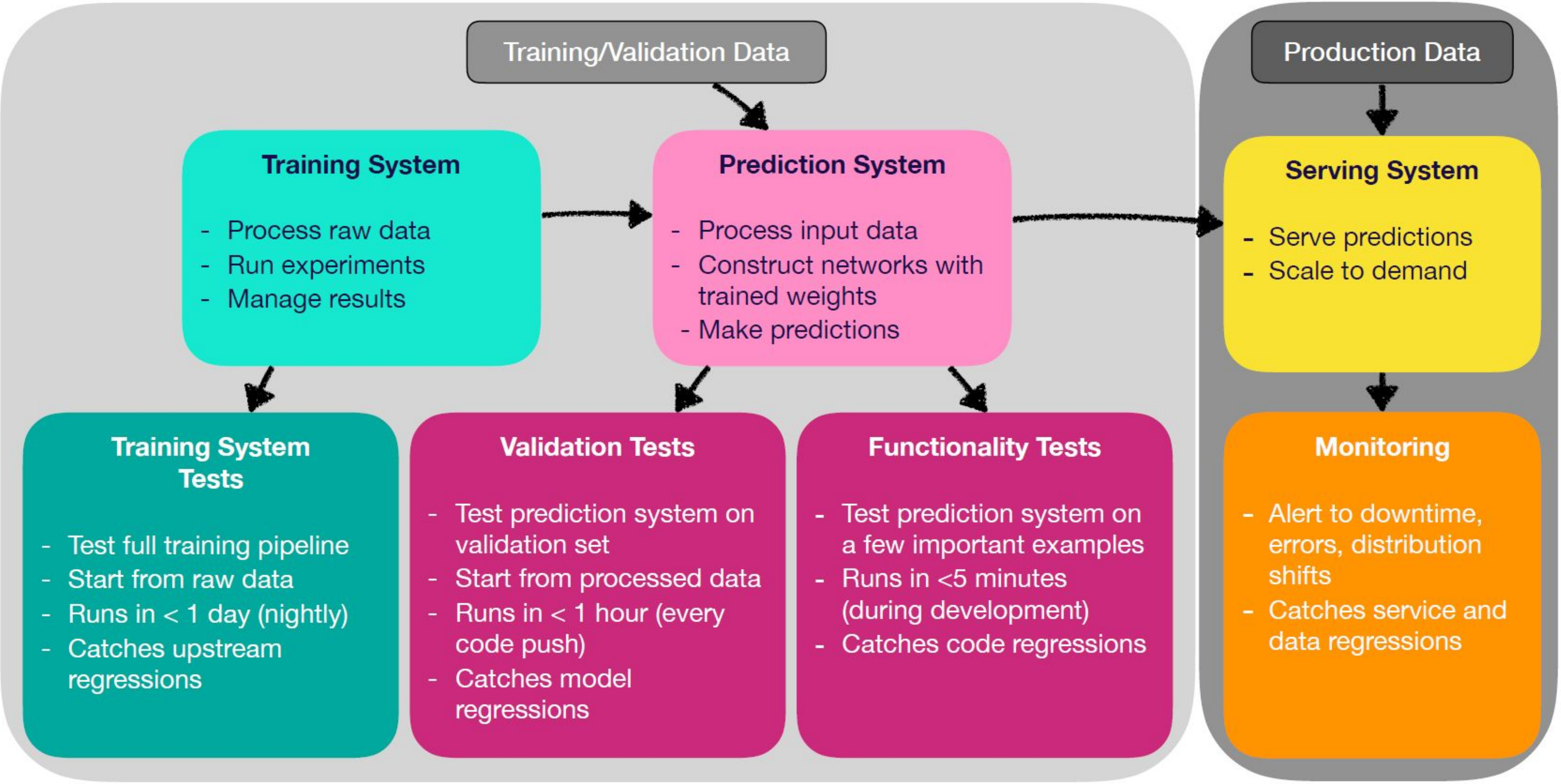
	[redacted]	**Production**
Objectives	[redacted]	Different stakeholders have different objectives
Computational priority	[redacted]	Fast inference, low latency
Data	[redacted]	Constantly shifting
Fairness	[redacted]	Important
Interpretability	[redacted]	Important

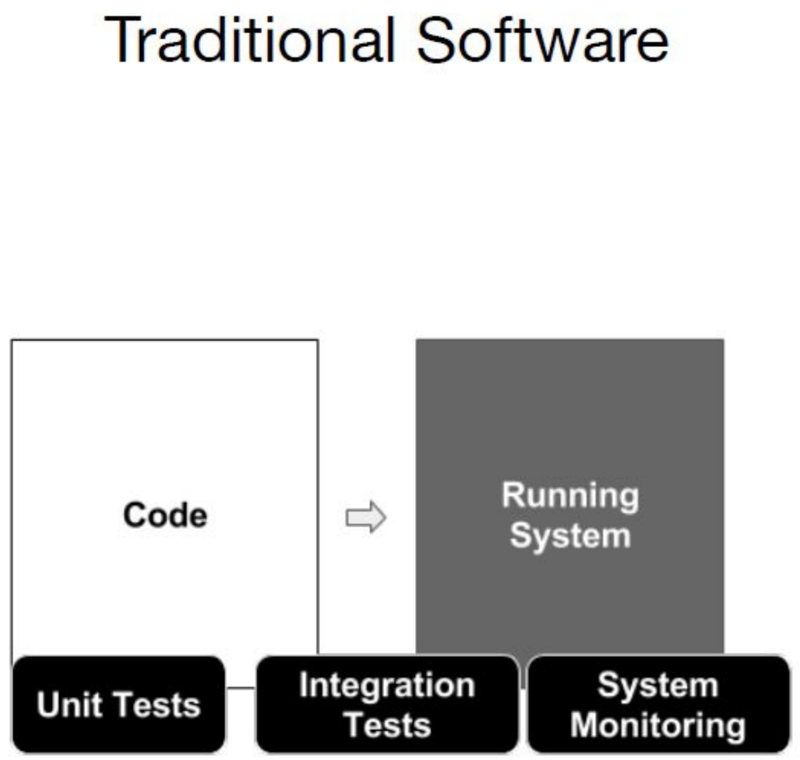
ML Ops Stack

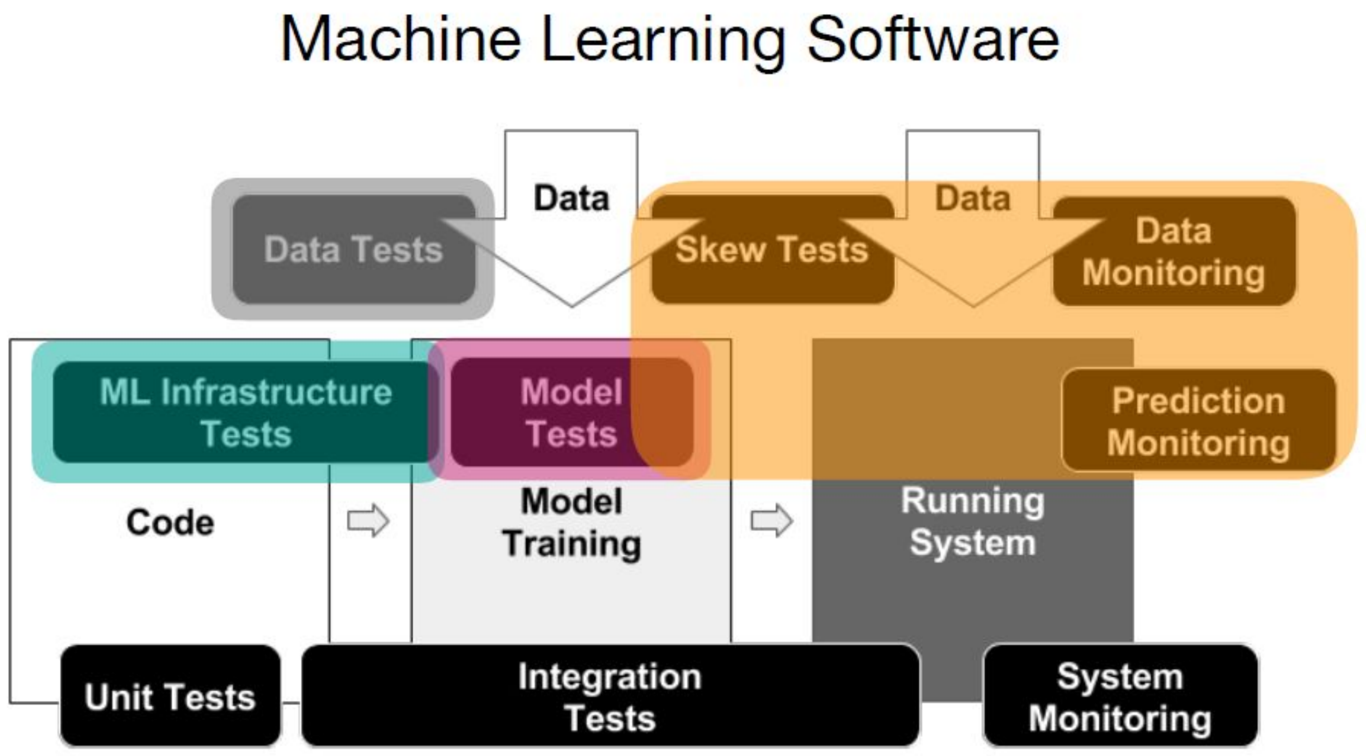

Amazon SageMaker
Determined AI
Neptune
H2O
FLOYD
DOMINO DATA LAB
"All-in-one"
Quilt
DVC
Pachyderm
LIQUIDATA
Versioning
figure eight
scale
prodigy
Labeling
PY
Luigi
Processing
Database
NFS
S3
Storage
or
Data
PYTORCH
RAY
Distributed Training
SIGOPT
tune
Hyperparameter Tuning
PYTORCH
K
fast.ai
Frameworks
comet
Weights & Biases
TensorBoard
LOSSWISE
Experiment Management
jupyter
git
Software Engineering
Polyaxon
PY
slurm
Resource Management
or
Development
Training/Evaluation
?
Monitoring
NVIDIA TensorRT
TensorFlow Lite
Hardware / Mobile
ALGORITHMIA
SELDON
Web
ONNX
Interchange
Buildkite
?
CI / Testing
or
or
Deployment

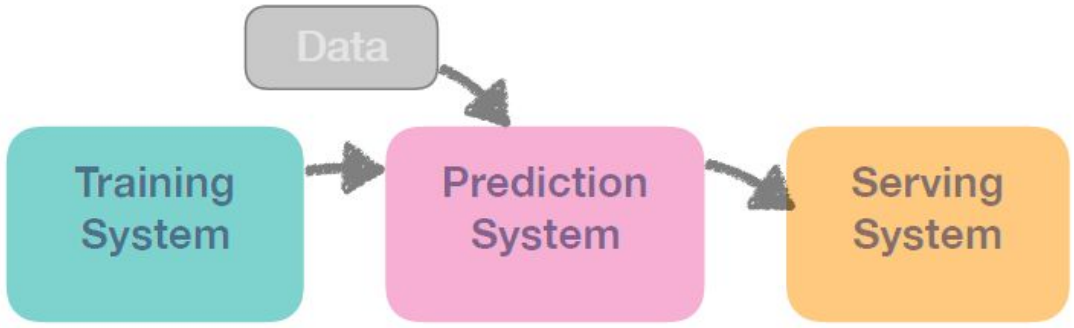
Testing ML Pipelines

Ref: https://fall2019.fullstackdeeplearning.com/


Training/Validation Data
Production Data
Training System
- Process raw data
- Run experiments
- Manage results
Prediction System
- Process input data
- Construct networks with trained weights
- Make predictions
Serving System
- Serve predictions
- Scale to demand
Training System Tests
- Test full training pipeline
- Start from raw data
- Runs in < 1 day (nightly)
- Catches upstream regressions
Validation Tests
- Test prediction system on validation set
- Start from processed data
- Runs in < 1 hour (every code push)
- Catches model regressions
Functionality Tests
- Test prediction system on a few important examples
- Runs in <5 minutes (during development)
- Catches code regressions
Monitoring
- Alert to downtime, errors, distribution shifts
- Catches service and data regressions


Traditional Software
Code
Running System
Unit Tests
Integration Tests
System Monitoring


Machine Learning Software
Data
Data
Data Tests
Skew Tests
Data Monitoring
ML Infrastructure Tests
Model Tests
Prediction Monitoring
Code
Model Training
Running System
Unit Tests
Integration Tests
System Monitoring


Data
Training System
Prediction System
Serving System

1	Feature expectations are captured in a schema.
2	All features are beneficial.
3	No feature's cost is too much.
4	Features adhere to meta-level requirements.
5	The data pipeline has appropriate privacy controls.
6	New features can be added quickly.
7	All input feature code is tested.

Data Tests

1	Model specs are reviewed and submitted.
2	Offline and online metrics correlate.
3	All hyperparameters have been tuned.
4	The impact of model staleness is known.
5	A simpler model is not better.
6	Model quality is sufficient on important data slices.
7	The model is tested for considerations of inclusion.

Model Tests

1	Training is reproducible.
2	Model specs are unit tested.
3	The ML pipeline is Integration tested.
4	Model quality is validated before serving.
5	The model is debuggable.
6	Models are canaried before serving.
7	Serving models can be rolled back.

ML Infrastructure Tests

1	Dependency changes result in notification.
2	Data invariants hold for inputs.
3	Training and serving are not skewed.
4	Models are not too stale.
5	Models are numerically stable.
6	Computing performance has not regressed.
7	Prediction quality has not regressed.

Monitoring Tests

Further Reading

Best Practices for ML Engineering:

https://developers.google.com/machine-learning/guides/rules-of-ml

References

- https://stanford-cs329s.github.io/index.html
- https://fall2019.fullstackdeeplearning.com/