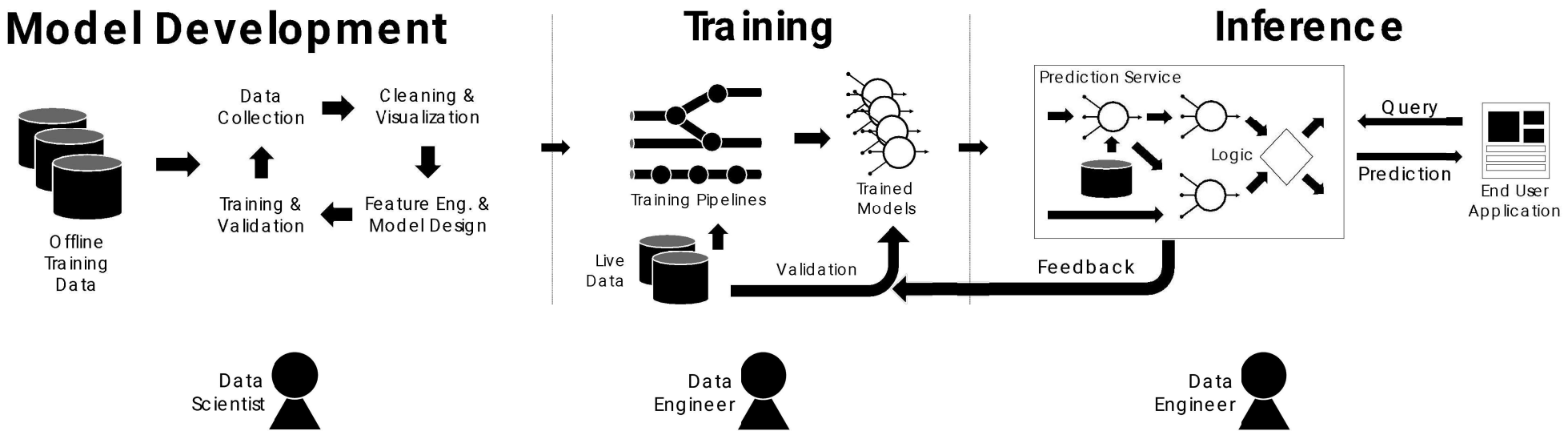
ML Ops
CSP7040

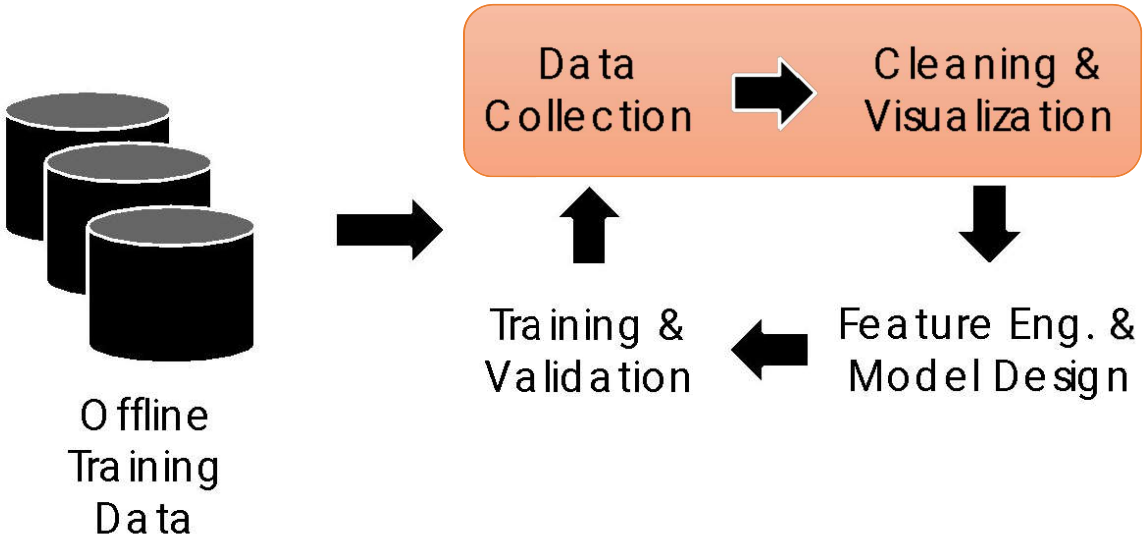
Adapted from the slides shared by Dr. Joseph E. Gonzalez of UC Berkeley for the course CS294 at https://ucbrise.github.io/cs294-ai-sys-fa19/

Machine Learning Lifecycle

What is the ***Machine Learning Lifecycle***?



Model Development



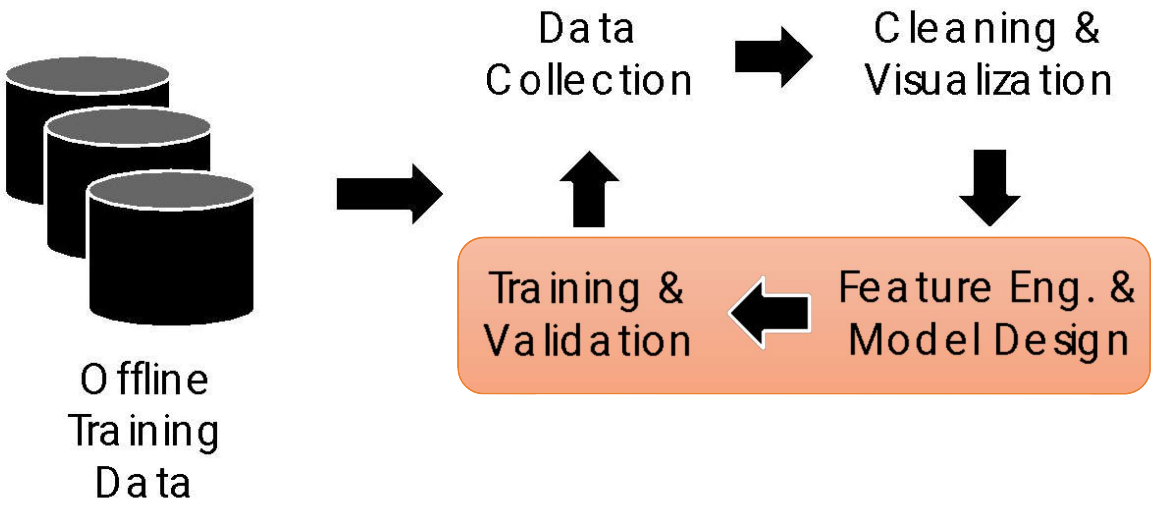
Identifying potential sources of data

Joining data from multiple sources

Addressing **missing values** and **outliers**

Plotting trends to identify **anomalies**

Model Development



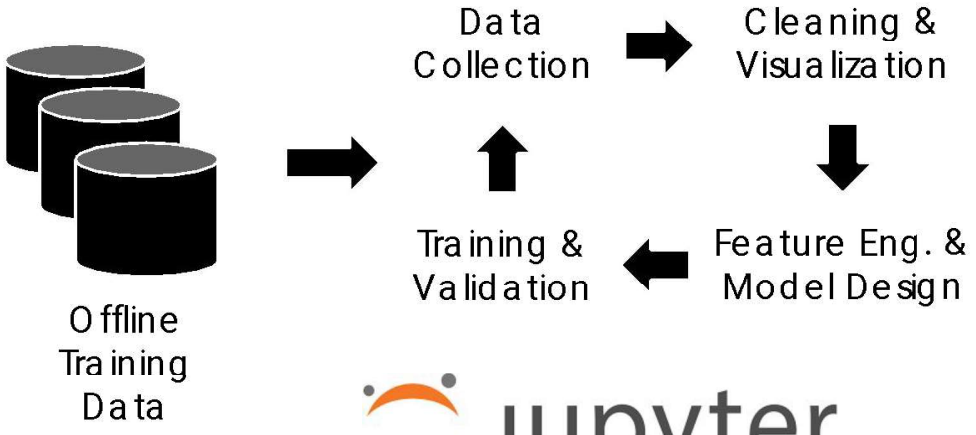
Building informative **features functions**

Designing new **model architectures**

Tuning hyperparameters

Validating prediction accuracy

Model Development *Technologies*



scikit learn

TensorFlow

R

PYTORCH

Keras

jupyter

mxnet

dmlc XGBoost

matplotlib

Caffe2

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

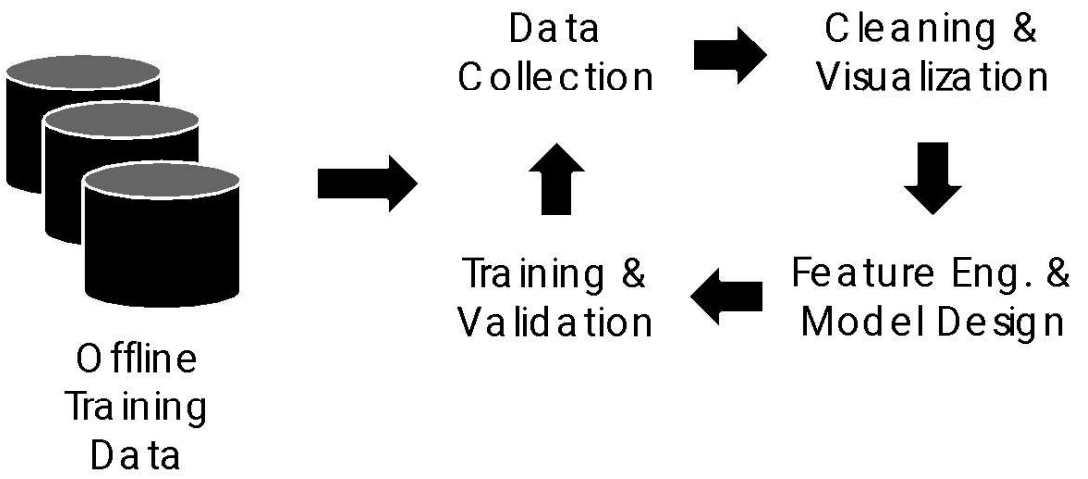
NumPy

APACHE Spark™

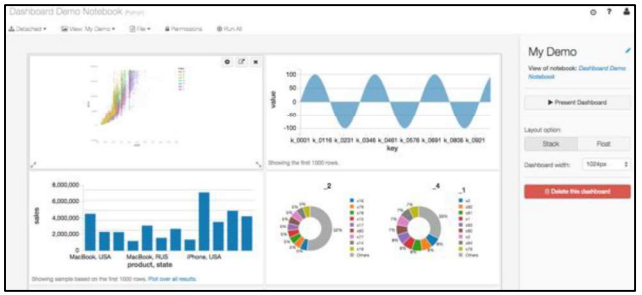
DASK

HIVE

What is the output of Model Development

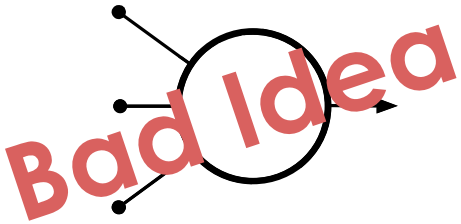


Reports & Dashboards



(insights …)

Trained Model

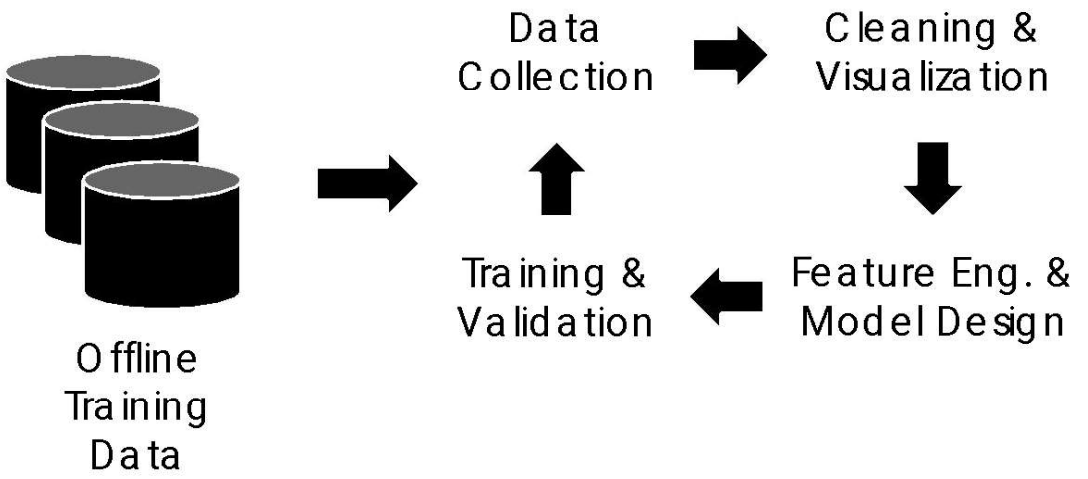


Why is it a **Bad Idea** to directly produce **trained models** from **model development**?

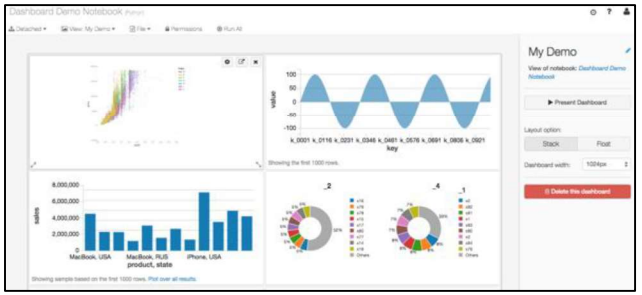
With just a trained model we are **unable to**

1. **retrain** models with new data
2. track data and code for **debugging**
3. capture **dependencies** for deployment
4. audit training for **compliance** (e.g., GDPR)

What is the output of Model Development

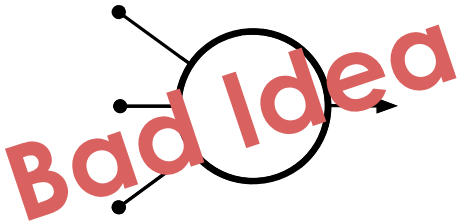


Reports & Dashboards

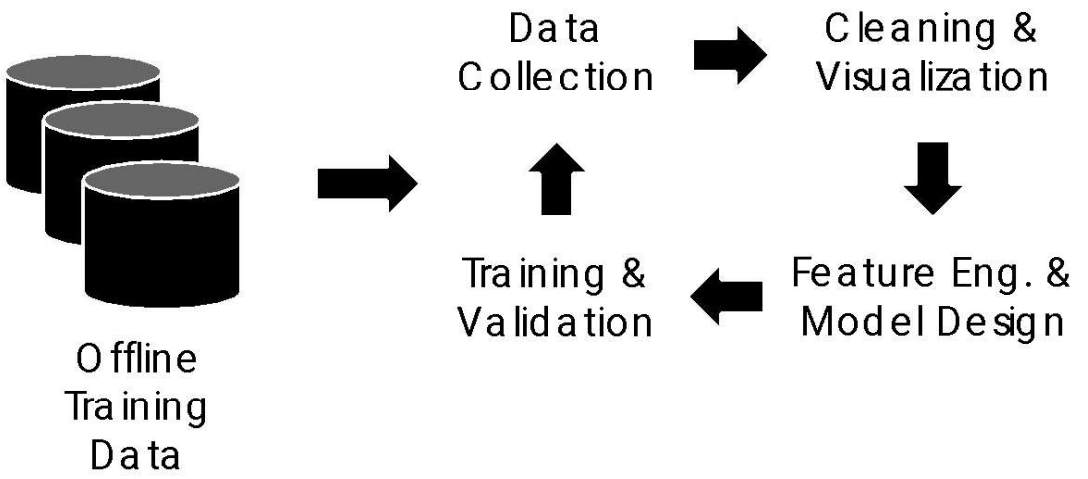


(insights …)

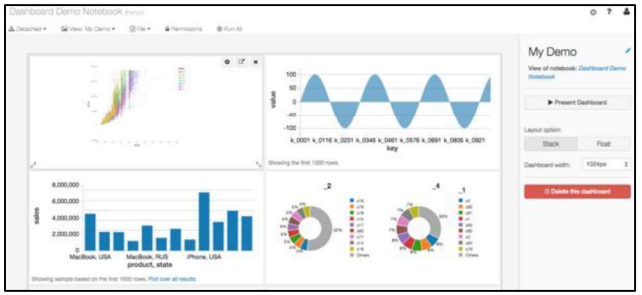
Trained Models



What is the output of Model Development

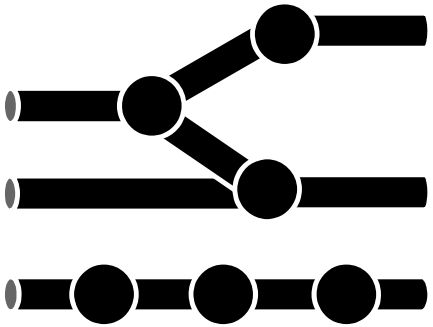


Reports & Dashboards



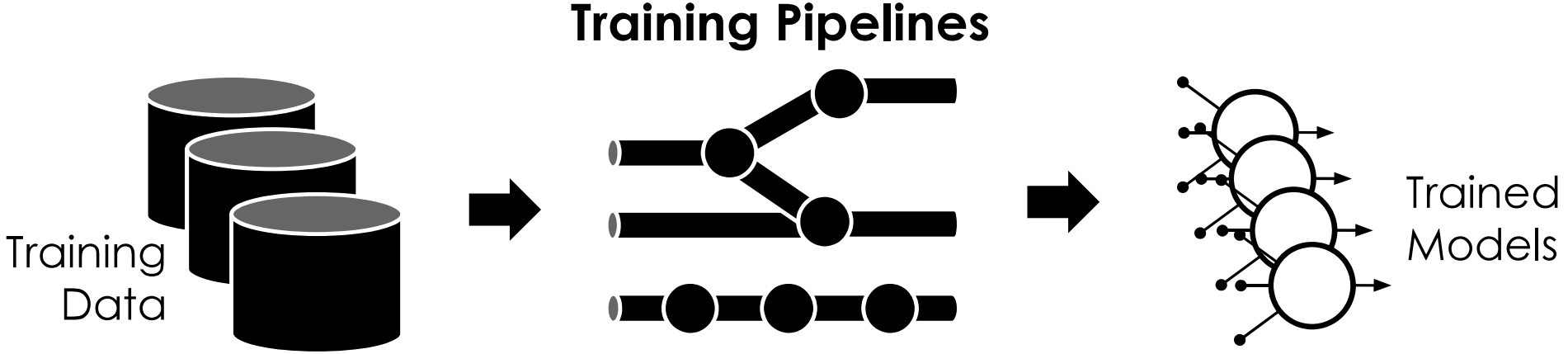
(insights …)

Training Pipelines



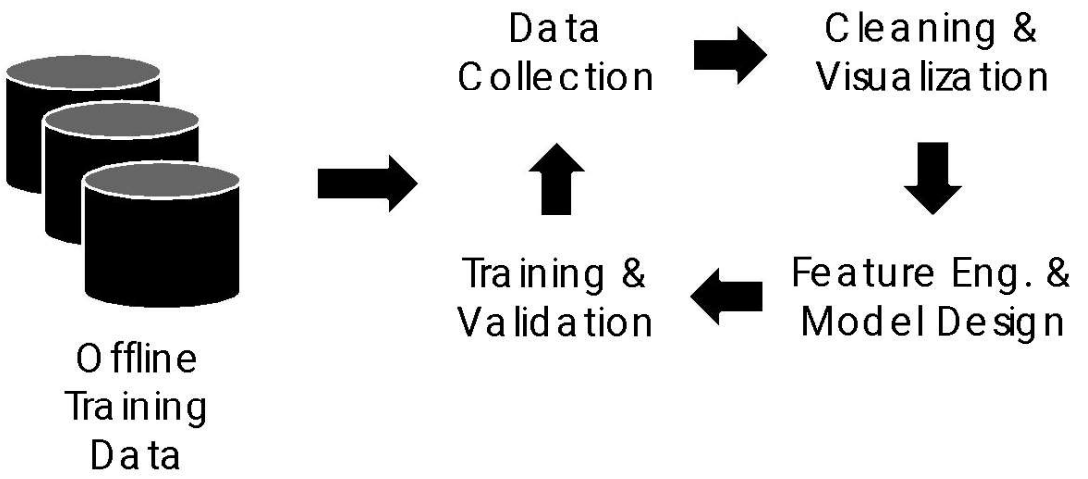
Training Pipelines Capture the **Code** and **Data Dependencies**

- Description of how to train the model from data sources

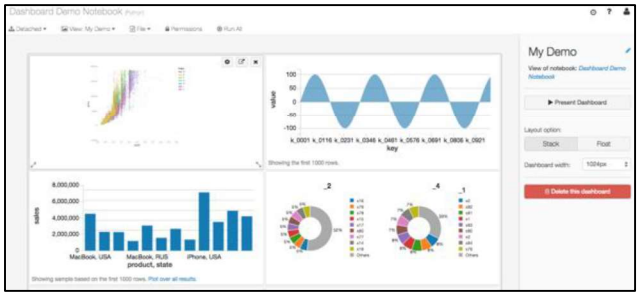


Software Engineering Analogy	Training Pipelines ≈ Code
	Trained Models ≈ Binaries

What is the output of Model Development

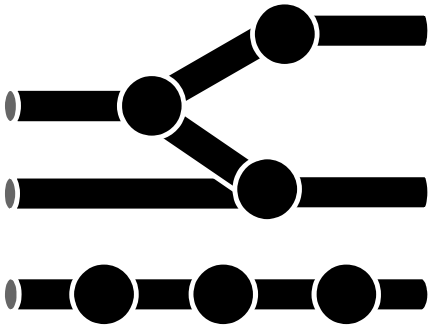


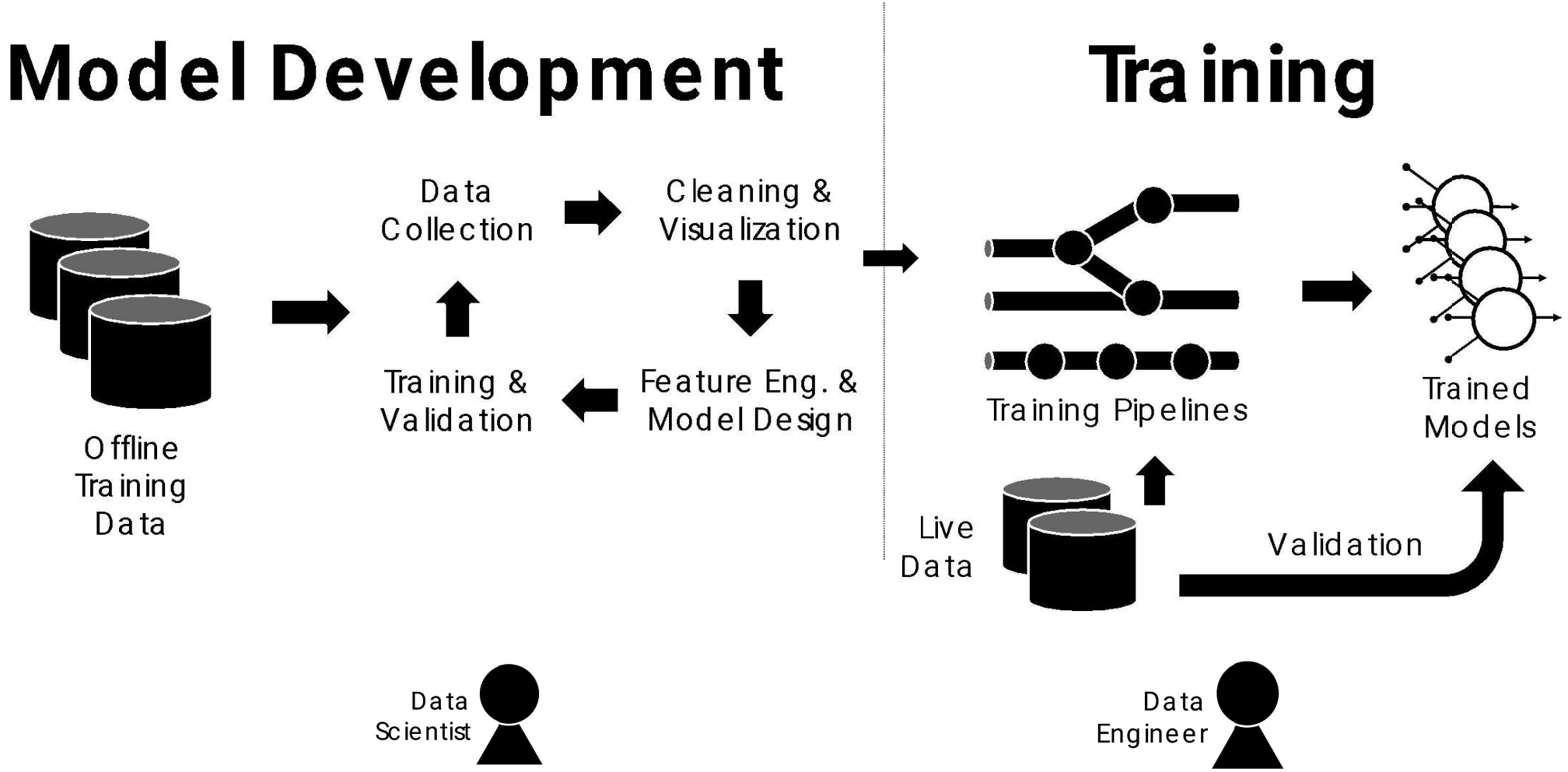
Reports & Dashboards



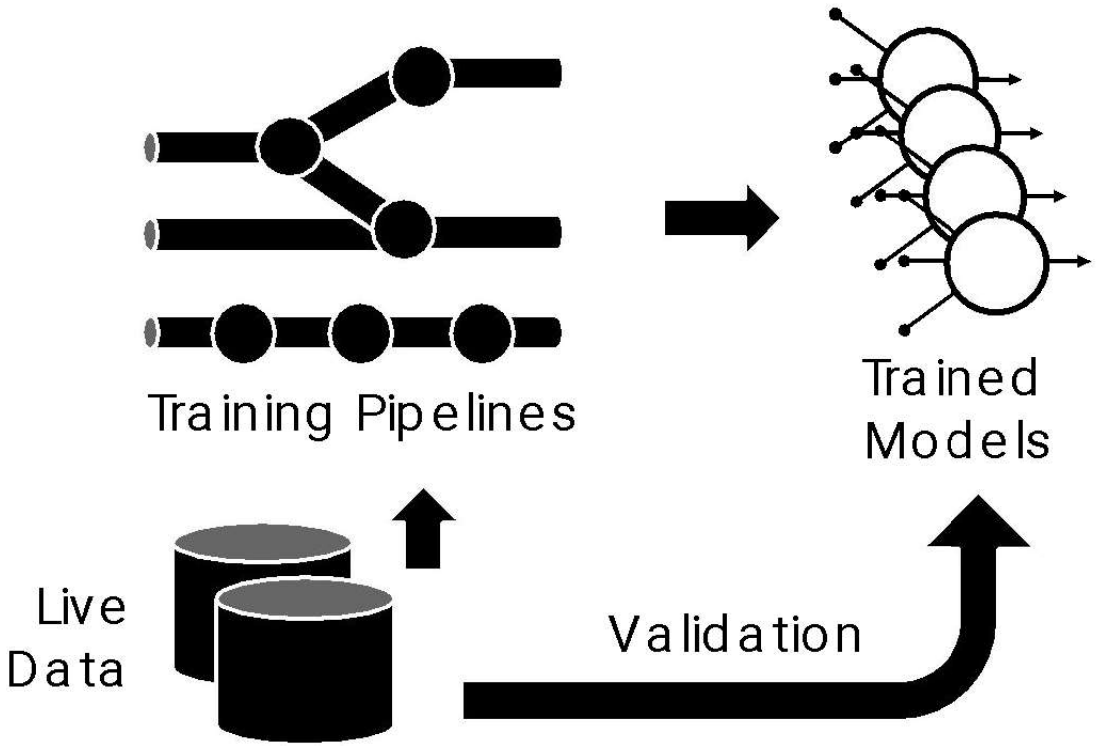
(insights …)

Training Pipelines




Model Development
Training
Data Collection
Cleaning & Visualization
Training & Validation
Feature Eng. & Model Design
Offline Training Data
Training Pipelines
Trained Models
Live Data
Validation
Data Scientist
Data Engineer

Training



Training models **at scale** on **live data**

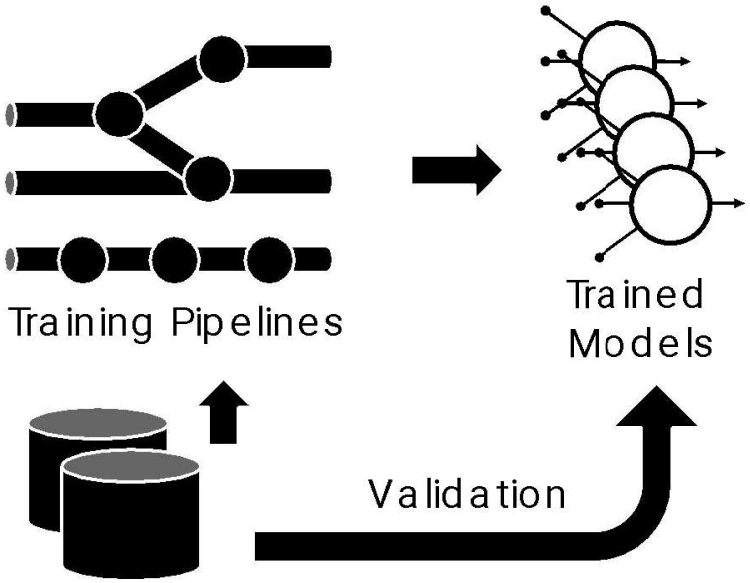
Retraining on new data

Automatically **validate** prediction accuracy

Manage model **versioning**

Requires **minimal expertise** in machine learning

Training *Technologies*



Workflow Management:

Apache Airflow

Luigi

Azkaban
Open-source Workflow Manager

OOZIE

Scalable Training:

PYTORCH

mxnet

TensorFlow

HOROVOD

APACHE Spark™

dmlc XGBoost

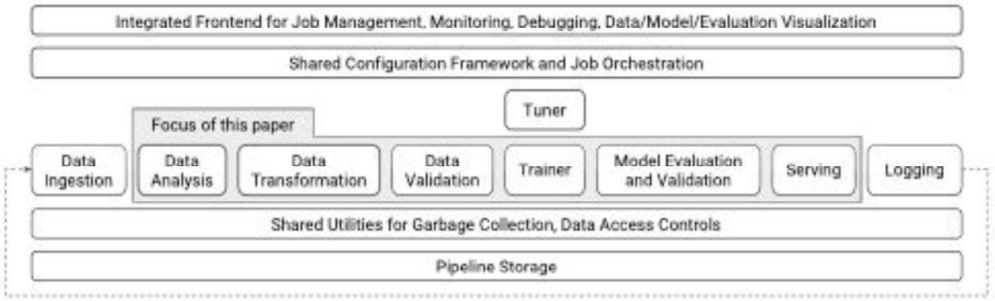
Suggested readings

- Hidden Technical Debt in Machine Learning Systems
  - NeurIPS'15, widely cited
  - Provides an overview of the challenges from Google
- TFX: A TensorFlow-Based Production-Scale Machine Learning Platform
  - KDD'17, now part of https://www.tensorflow.org/tfx (sort of)
  - Googles solution to the challenges in the first paper
- Towards Unified Data and Lifecycle Management for Deep Learning
  - ICDE'17, Video Demo
  - An alternative database community solution

What to think about when reading

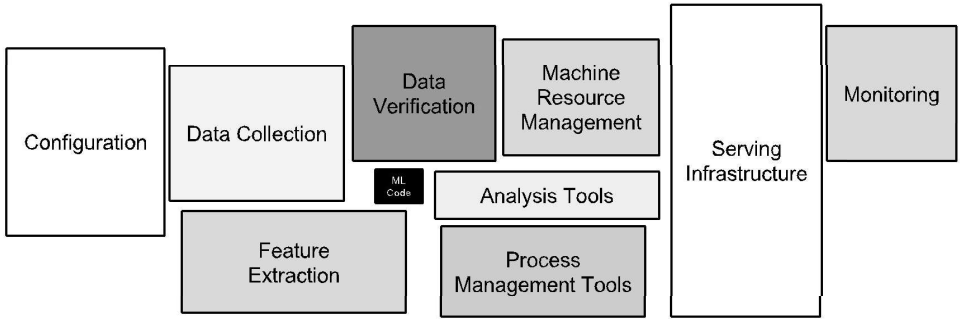
- How does the work differentiate between engineering and research challenges?
- What innovations in machine learning are needed?
- What are the key research challenges proposed and addressed?
- Are the proposed solutions too opinionated
  - Would they require top down mandates for adoption?
  - Would you use these systems?
  - Are they sufficiently flexible to support innovation

TFX: A TensorFlow-Based Production-Scale Machine Learning Platform



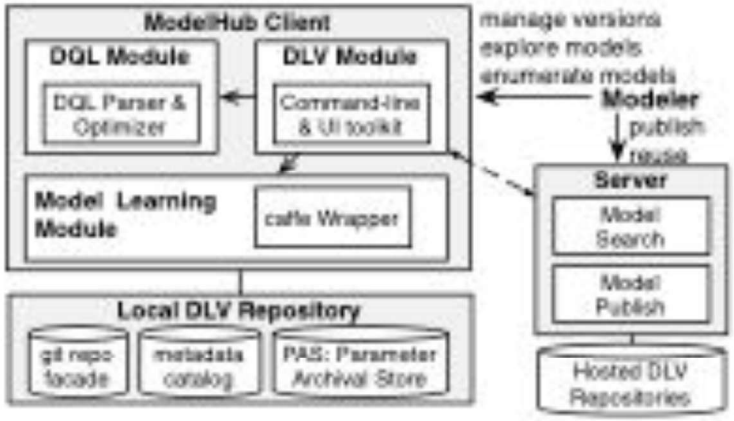
- Describes solutions to many of the problem outlined in the technical debt paper.
- **Key Idea:** Adapt best practices for software development to address machine learning lifecycle
  - empathetic to the *reality* of "machine learning developers"
- **Contributions:** actual system, interesting ideas around data and model validation, schema enforcement, and meaningful errors.

Hidden Technical Debt in Machine Learning Systems



- **Technical Debt:** long term development and maintenance costs incurred by expedient design decisions
- **Key Idea:** machine learning deployments often incur substantial technical debt (compared to traditional software)
- **Contribution:** this paper characterizes the forms of technical debt and alludes to possible compensating actions

Towards Unified Data and Lifecycle Management for Deep Learning



- Describes a system (ModelHub) for managing, querying, and manipulating models and their related metadata.
- **Key Idea(?):** Model lifecycle management combines code and data (parameters) → a natural API would then combine version control commands with SQL-like querying.
- **Solution:** Combines a git-like client API with a SQL-like querying interface to enable basic actions and more complex queries.
  - Leverages optimizations to store model weights more efficiently.
    - (necessary?)

Related Systems Efforts

- Doing Machine Learning the Uber Way: Five Lessons From the First Three Years of Michelangelo
- Introducing FBLearner Flow: Facebook's AI backbone
- KubeFlow: Kubernetes Pipeline Orchestration Framework
- DeepBird: Twitters ML Deployment Framework
- Mlflow: A System to Accelerate the Machine Learning Lifecycle
- Data Engineering Bulletin on the Machine Learning Lifecycle
  - Full disclosure: I was the editor