Machine Learning Model Management and Inference (MLMMI)
01 Introduction and Overview

Dr. Arnab Phani
BIFOLD, TU Berlin
DEEM Lab

About ME

- **About Me**
  - Since 07/2025: **Postdoc** at DEEM Lab
  - **System infrastructure for ML**, compiler, multi-backend runtime
  - Data management for data science
  - 2019-2024: **PhD from DAMS Lab** under Prof. Matthias Boehm
  - Fine-grained reuse and feature transformations in machine learning systems
  - PMC member of **Apache SystemDS**
  - Prior to 2019: Senior Software Engineer at **Teradata Labs**

- **DEEM Lab (Led by Prof. Sebastian Schelter)**
  - Fundamental research at the **intersection of data management and ML**
  - **Lower the technical bar** for reliably working with AI technologies
  - Focus Areas: automated data validation, data-centric debugging, data processing in compliance with legal regulations, such as the "**right-to-be-forgotten**"
  - Primary publication venues: SIGMOD, VLDB and other top-tier venues.

Agenda

- **Course Organization**
- **Course Motivation and Goals**
- **Stratum System Overview**
- **Course Outline, Exercises, and Project**

Course Organization

Course Logistics

- **Staff**
  - **Lecturer:** Dr. Arnab Phani
  - **Teaching Assistant:** M.Sc. Elias Strauss

- **Language**
  - **Lectures and slides:** English
  - **Communication with TA:** English/German

- **Course Format**
  - VL/UE, 6 ECTS, **maximum capacity is 24**
  - Mandatory three programming assignments, preparation of questions, and group projects
  - Weekly lectures (Wed 16.15 in **FR 710** / E-N 189 in-person)
  - Weekly (optional) consultation with Elias (Mon 16.15 in MAR 0.001)
  - Recommended reading

Course Logistics, cont.

- **Prerequisites**
  - Completed bachelor degree
  - Applied ML and data management courses

- **Websites**
  - https://isis.tu-berlin.de/course/view.php?id=47383 (TUB-internal) ⟸ Updates and announcements
  - https://phaniarnab.github.io/courses/mlmmi_ss2026.html (public)

- **Academic Honesty**
  - No plagiarism
  - LLMs for coding is permitted but **not recommended**

- **Disclaimer**
  - First iteration: hiccups expected
  - Immediate feedback encouraged

Course Admission

- **Selection Process**
  - Buckets of different student categories (e.g., elective, additional module)
  - Fill up the form by **Friday (17/04) 18.00**.

- **Form Password**

mlmmi-ss26#deem

Course Motivation and Goals

Real-World ML

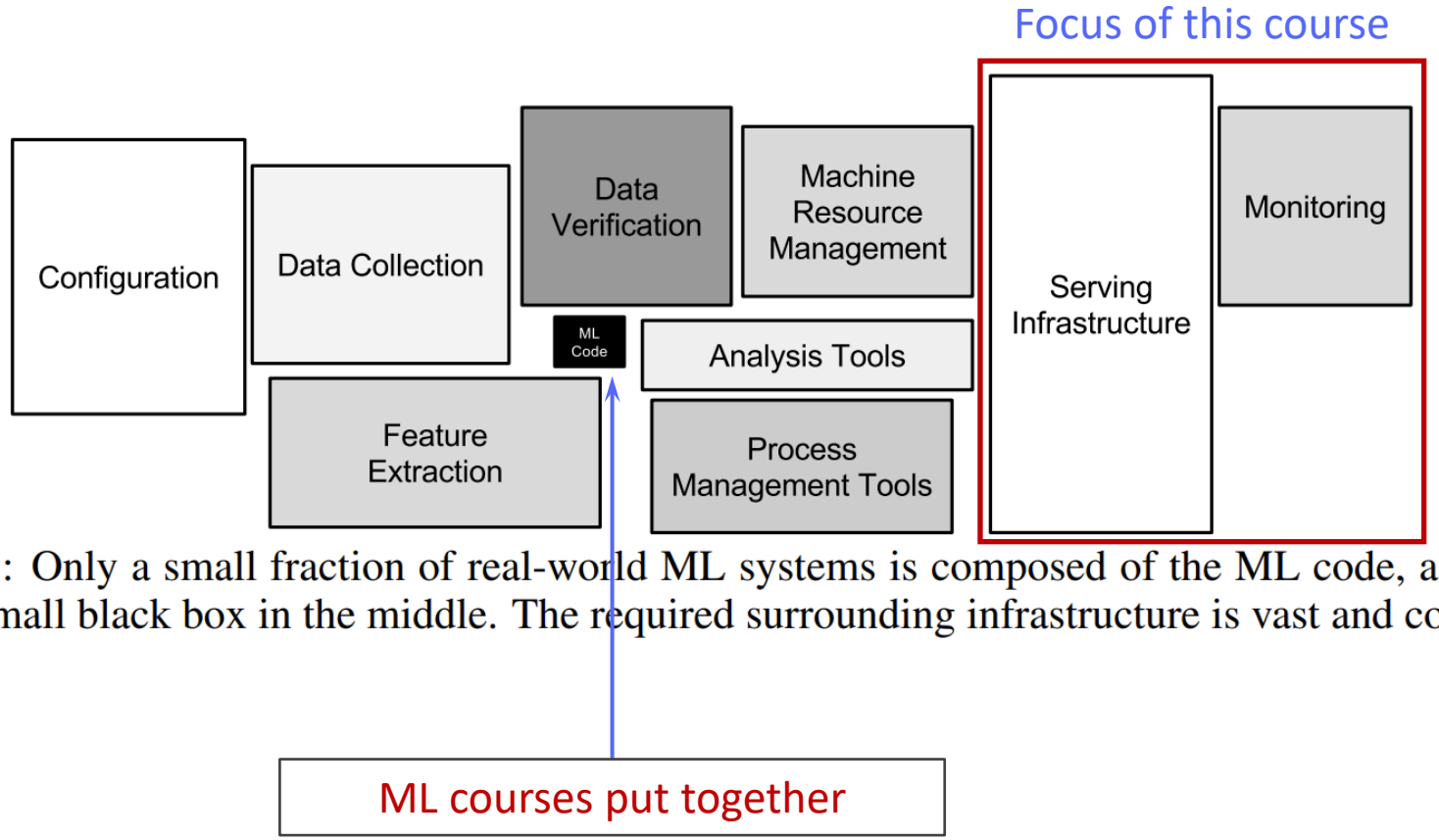
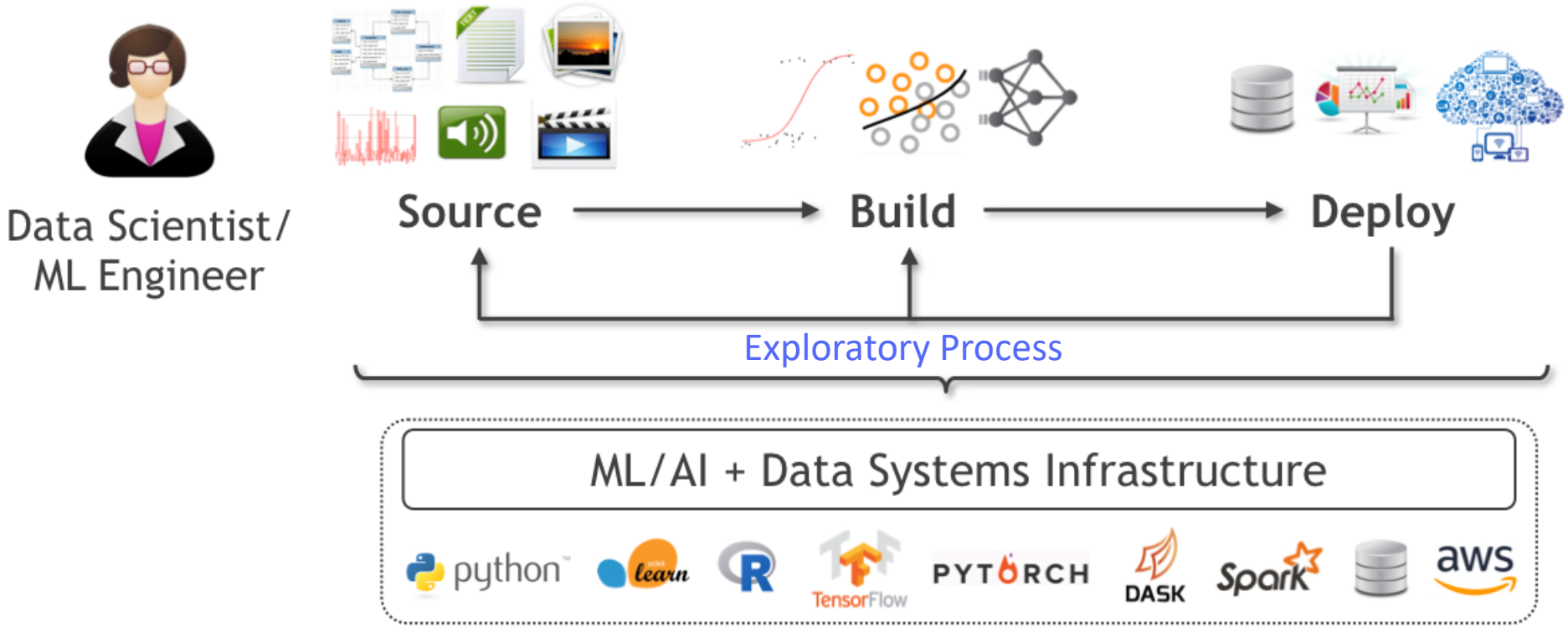


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Recap: Data Science Lifecycle



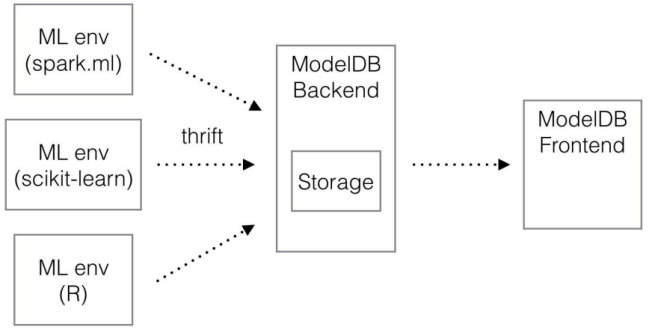
Data Integration
Data Cleaning
Feature Transformation

Model Selection
Model Training
Hyper-parameter Tuning

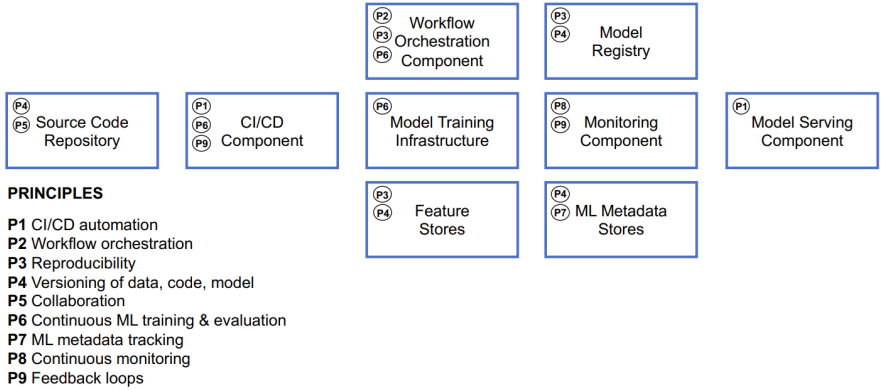
Deployment and Inference
Model Serving
Monitoring and Versioning

Course Goals

- **Model Management**
  - Data scientists build hundreds of models
  - Iterative and ad-hoc
  - Model Management: Tracking, storing, and indexing ML models



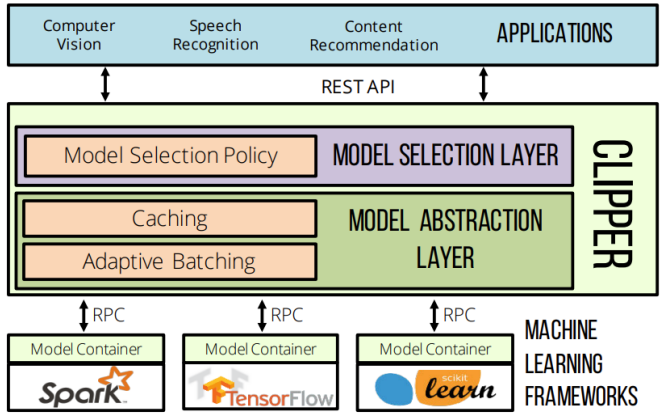
- **ML Platform**
  - Support for continuous training and serving
  - Human-in-the-loop
  - Deployment and monitoring
  - MLOps: DevOps for ML-infused software



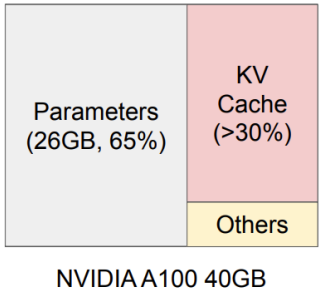
- **Model Selection**
  - Compare hundreds training configurations
  - Hyperparameter tuning
  - Algorithm/architecture selection

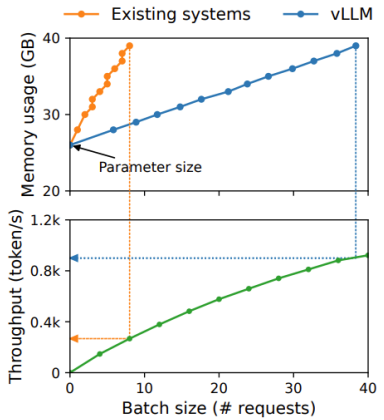
Course Goals, cont.

- **Model Serving Systems**
  - Real-time, more queries than training
  - Low latency, high throughput
  - Caching, batching, quantization, MQO



- **LLM Serving**
  - Autoregressive token generation
  - Efficient memory management
  - Token-level, model-level, system-level KV cache optimization





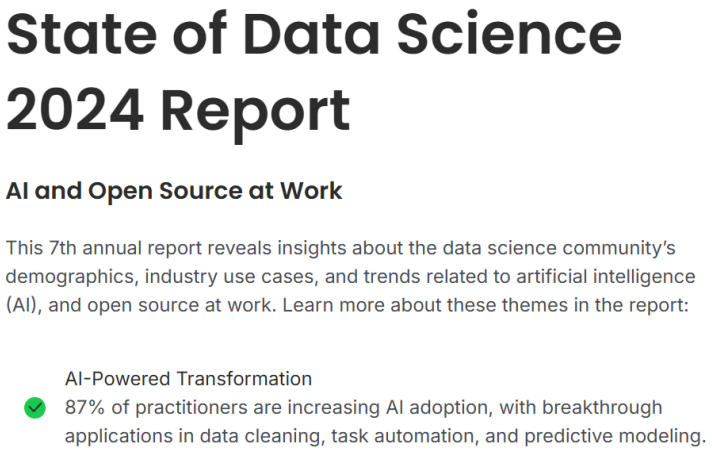
- **Compound AI Systems and Agents**
  - RAG: indexing, vector stores and search
  - LLM for data management, data management for agents
  - ML engineering agents

Why Take This Course?

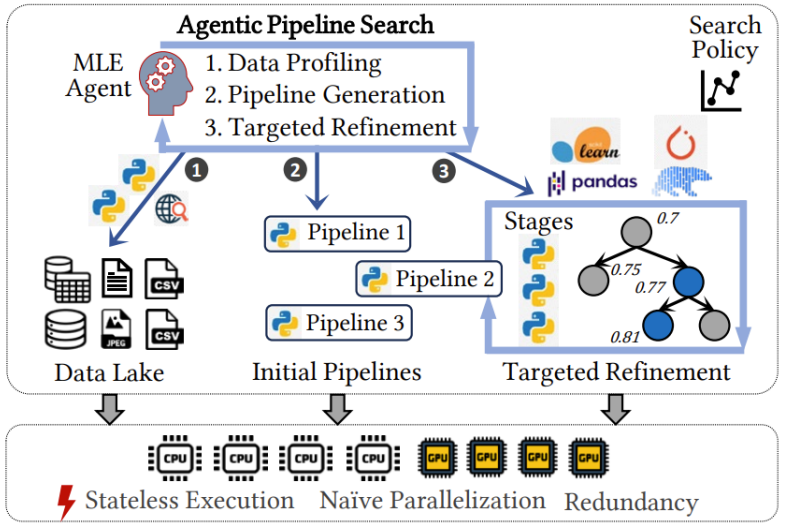
- Want to understand what's going on **under the hood** of serving platforms
- Interest in **data management**/systems
- Enjoy learning from **research papers** and discussions
- Enjoy implementing research papers and projects

stratum: A System Infrastructure for Massive Agent-Centric ML Workloads

Agentic Workloads

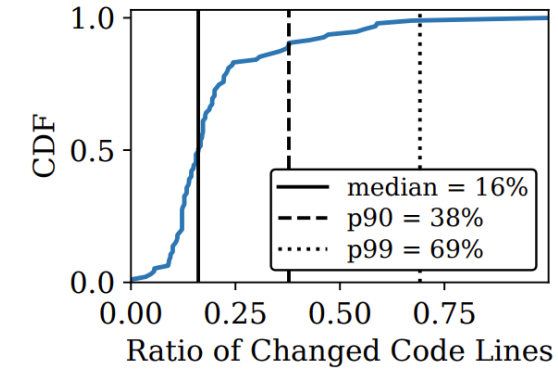


- **LLM-assisted Data Science**
  - **ML tasks as code optimization** problems
  - Explore many candidate solutions
  - Fully/semi-autonomous agents and AI-assisted programming
  - Data profiling, pipeline generation, targeted refinement; repeat
  - Most code in the future will be generated by **agents**



Challenges of Agentic Pipeline Search

- **Agentic Pipeline Search**
  - Large numbers of heterogeneous Python code
  - Overlapping executions, inefficient resource utilization, OOM failures, …
- **Insufficient ML Libraries**
  - Python ML ecosystem
  - ML libraries lack optimizing execution layer
  - Python is not designed for performance
- **Lack of Efficient Systems for Data Science**
  - Systems research community made advances
  - DSL-based systems (SystemDS, OptiML, DAPHNE)
  - Program compilation, cost-based optimization, multi-backend runtime
  - Lack of open code for LLMs to train on
  - Workload-specific systems (Polars, XGBoost, PyTorch)
  - Cannot execute end-to-end data science pipeline



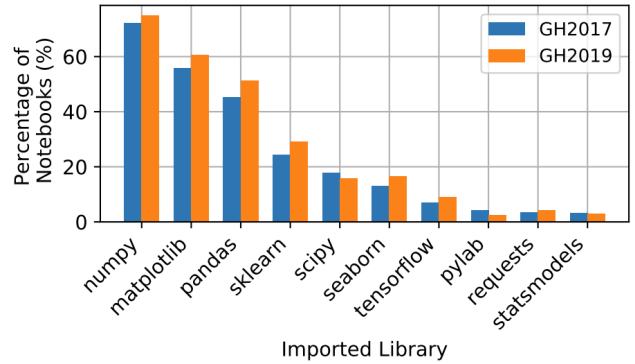


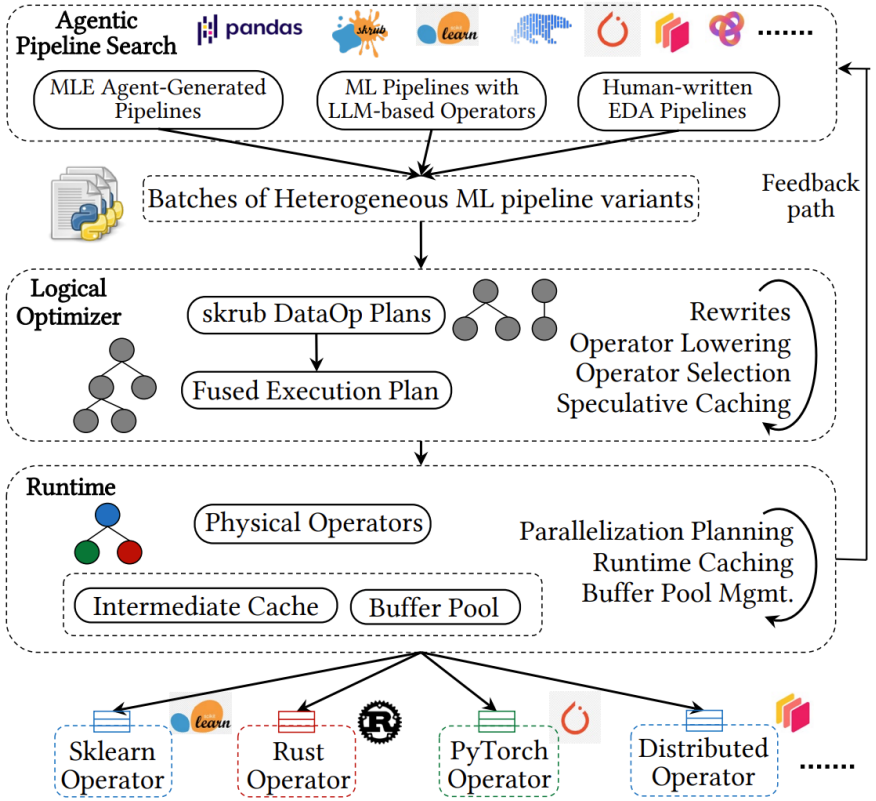
Figure 2. Top-10 used libraries across notebooks.

Apache SystemDS ™

Data Science Through the Looking Glass: Analysis of Millions of GitHub Notebooks and ML.NET Pipelines. SIGMOD Record. Vol 51 (2)

stratum: System Architecture

stratum



A DAG makes program structure explicit, allowing automatic analyze and optimization

Apply (algebraic) rules to rewrite to equivalent but efficient plan

Choose optimal physical implementation and execution details (where and how)

- **stratum**
  - A case for a new system
  - A foundation for agent-system co-design

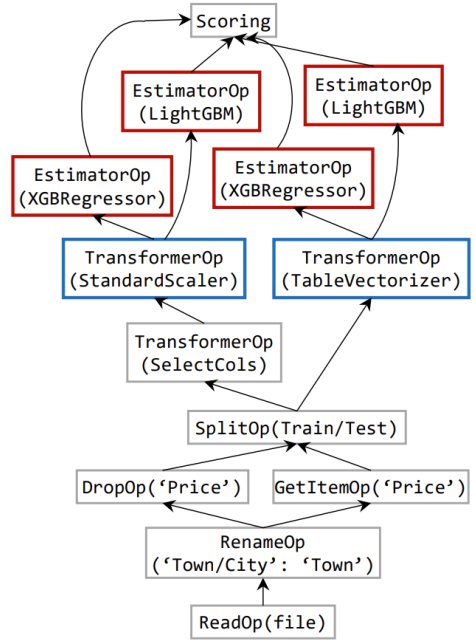
Consider starring the github repo:
https://github.com/deem-data/stratum

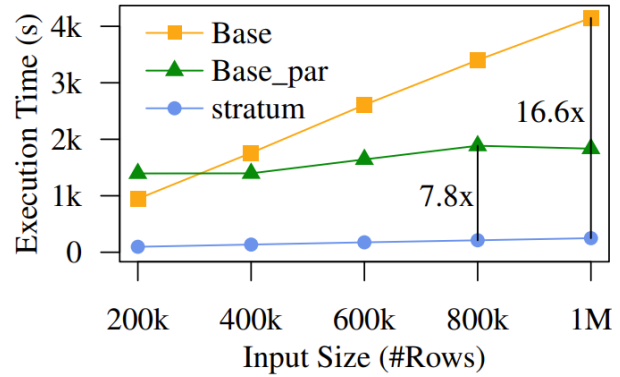
stratum: A System Infrastructure for Massive Agent-Centric ML Workloads [Vision]. CoRR abs/2603.03589

stratum Components

- **Declarative Abstraction**
  - Optimization requires operator semantics
  - Stratum adopts skrub DataOps
- **Logical Optimizer**
  - Metadata collection
  - Classical and API-aware rewrites
  - Operator lowering and selection
- **Runtime**
  - Rust backend
  - Parallelization planning: inter- and intra-operator parallelism
  - Reuse of intermediates
- **Open Challenges**

```
prd_c = prd_c.skb.apply(OneHotEncoder())
prd_n = prd_n.skb.apply(StandardScaler())
prd_vec = prd_c.skb.concat([prd_n])
X_jn = X.merge(prd_vec, …) #join
X_jn.drop(columns=["ID", "basket_ID"])
model = LGBMRegressor(random_state=42)
preds = X_jn.skb.apply(model, y=y)
search = preds.skb.make_grid_search(cv=cv)
```





Course Outline and Exercises/Projects

Course Outline (Tentative)

- Model Management Systems [Apr 22]
- ML Platforms and Deployment [Apr 29]
- Model Selection Systems [May 06]

- ML Serving Systems [May 20]
- LLM Serving 1
- LLM Serving 2
- MLE Agents

- Exercise 1 Submission [May 04]
- Exercise 2 Submission [May 18]
- Exercise 3 Submission [Jun 01]

- Invited Talk of **Nils Strassenburg** [May 13]
- Invited Talk 2
- Questions for Invited Talk 1 [May 11]
- Questions for Invited Talk 2

- Project Pitch [Jun 08/10]
- Project Presentations

Exercises

- **Exercise 1: Model Management System**
  - Dataset store and versioning
  - Model registry and lineage tracking
  - Model selection under fixed budget

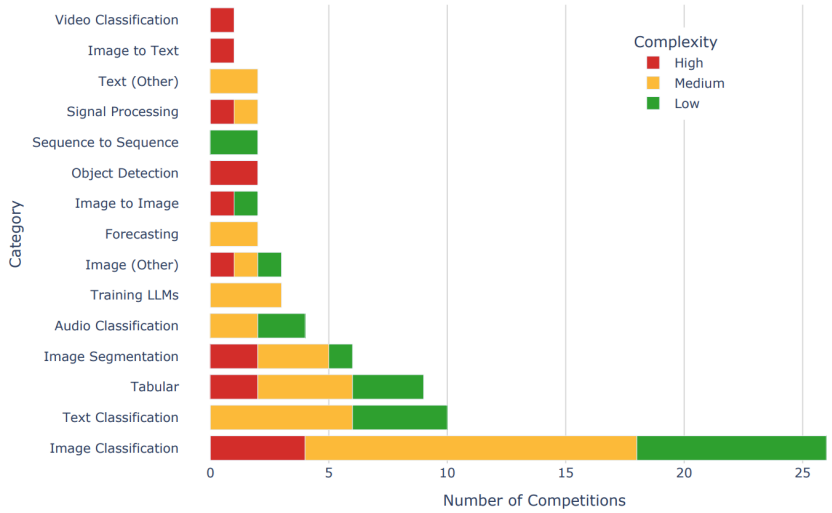
```
1_Model_registry_and_selection/
├── requirements.txt
├── solution.ipynb            # main notebook orchestrating the pipeline
├── data/
│   ├── raw/                  # original dataset
│   └── ds_v*/                # versioned datasets (train.csv, test.csv, metadata.json)
├── models/
│   └── model_*/              # serialized model artifacts (e.g. .joblib)
├── registry/
│   └── model_*.json          # one metadata file per registered model
└── scripts/                  # reproducible Python scripts
    ├── create_datasets.py
    ├── train_models.py
    ├── select_model.py
    └── summarize.py
```

- **Exercise 2: Transfer Learning**
  - Small model and dataset
  - Freeze first few layers, fine-tune rest
  - Compare full training, full fine-tuning, transfer learning configurations
  - Apply optimizations

- **Exercise 3: Model Serving Systems**
  - Local serving system with request queue
  - Apply optimizations such as batching and prediction caching
  - Report throughput and latency
  - Connect with the model registry from Exercise 1

Example Project Categories

- **MLE-bench (75 Kaggle Competitions)**
  - Write 5-10 scripts using skrub DataOps
  - Adapt an MLE agent to generate skrub code
  - Other Kaggle competitions for tabular data
- **Skrubify Agent/Compound AI System**
- **Features for Stratum**
  - API configuration for running server
  - Text-based execution plan
  - Logical rewrites and runtime optimizations
- **Rust Kernels**
  - Popular ML operators and algorithms
  - Performance benchmark and unit tests



	Top Transformers	Top Learners
SKLEARN	StandardScaler CountVectorizer TfidfTransformer PolynomialFeatures	LogisticRegression MultinomialNB SVC LinearRegression RandomForestClassifier

Request API: ChatAI
https://kisski.gwdg.de/en/leistungen/2-02-llm-service/

Data Science Through the Looking Glass: Analysis of Millions of GitHub Notebooks and ML.NET Pipelines. SIGMOD Record. Vol 51 (2)

Summary

- Model management is a data management problem
- Model serving requires system-level optimizations
- Stratum is a new cool system
- Research focused exercises and projects