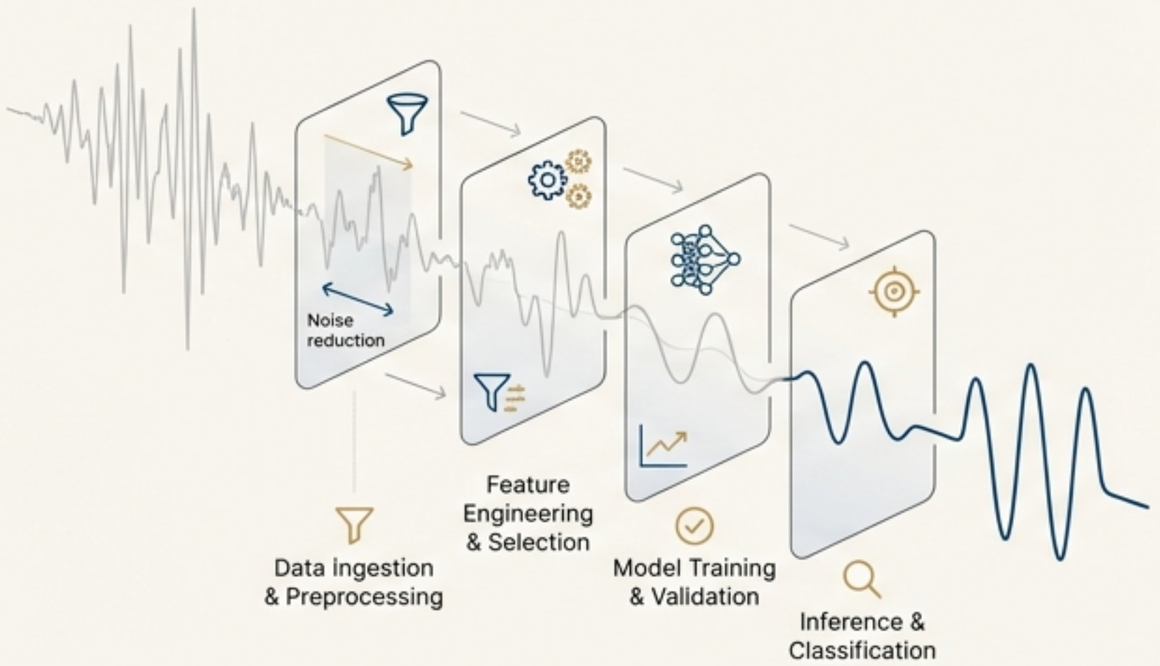
Deconstructing the MLOps Pipeline for Human Activity Recognition

A technical walkthrough of the end-to-end system for recognizing anxiety-related activities from wearable sensor data.



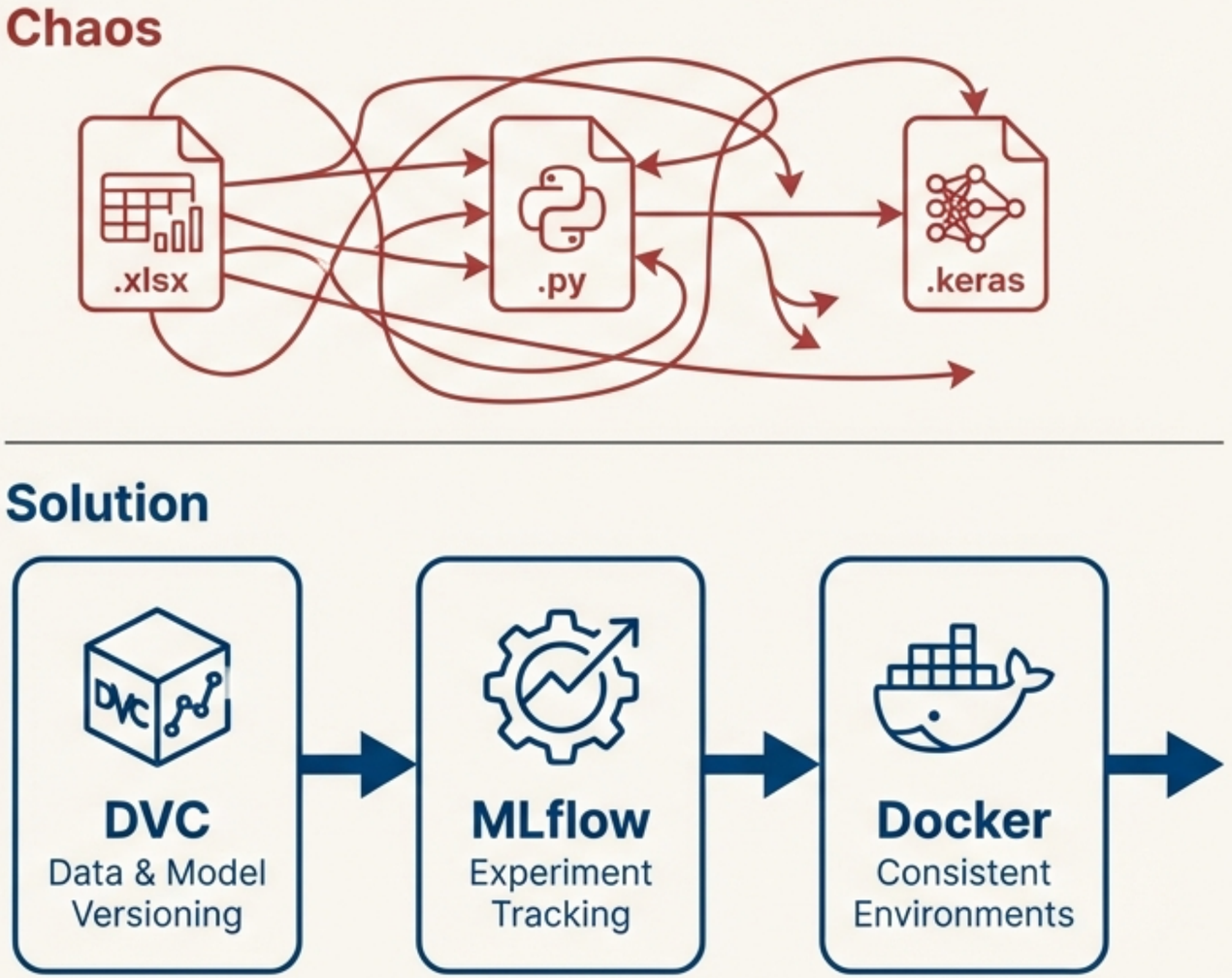
1D-CNN-BiLSTM Architecture

Garmin Smartwatch Sensors

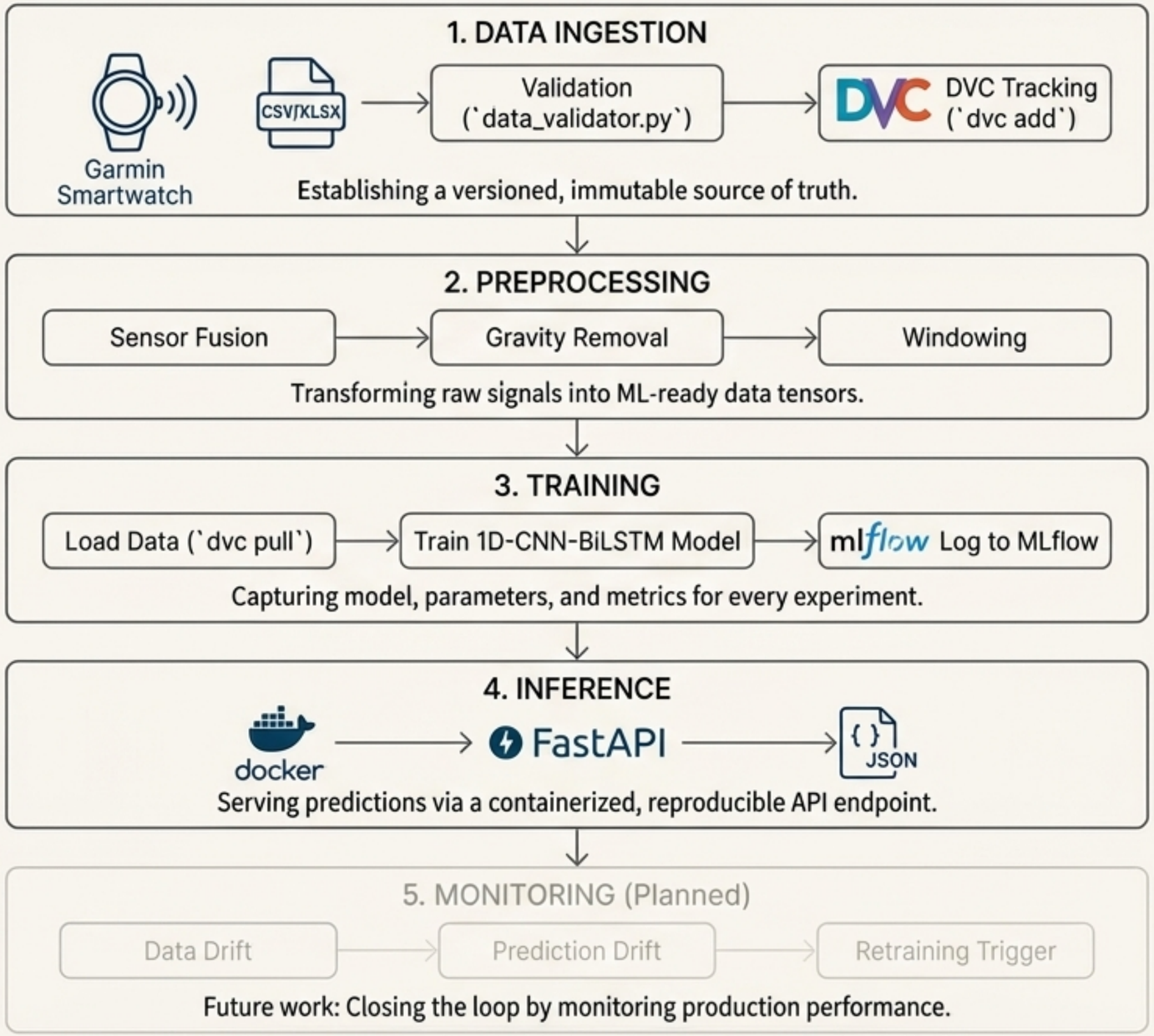
DVC mlflow docker

The MLOps Imperative: From Data Chaos to Reproducible Insight

- **Data & Model Versioning:** How do we track massive datasets and the models trained on them without cluttering our Git history?
- **Experiment Traceability:** How can we reliably reproduce results and compare which training parameters led to the best model?
- **Deployment Consistency:** How do we eliminate the “it works on my machine” problem and ensure our model performs identically in production?



The End-to-End Blueprint



Stage 1: Ingestion & Versioning the Source of Truth

Input → **Process** → **Output**

*.xlsx (Garmin exports)
`data/raw/`

data_validator.py

Validates schema, checks for NaNs, and ensures data quality before it enters the pipeline.

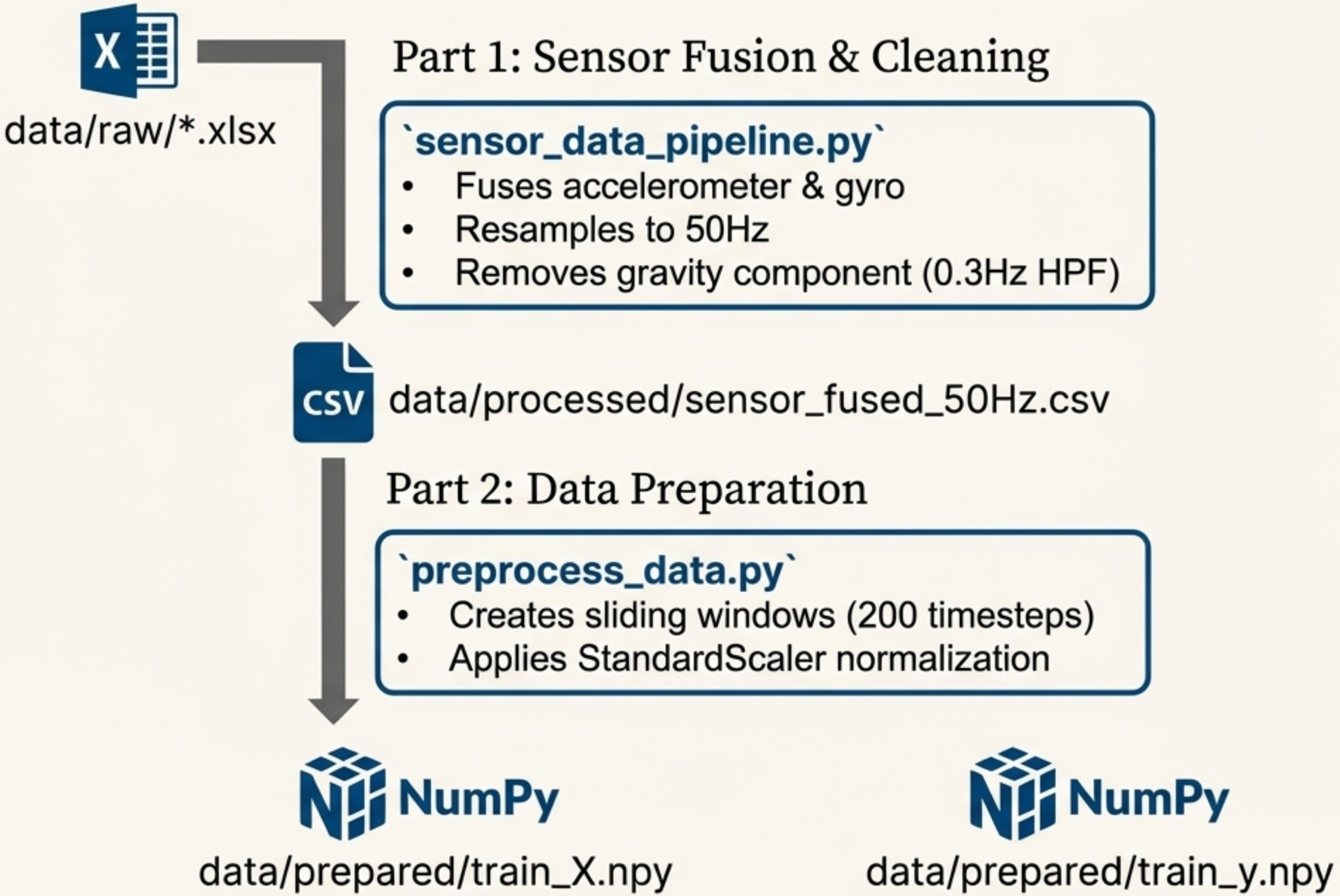
Validated Data
`data/raw/`

Tool Spotlight: DVC

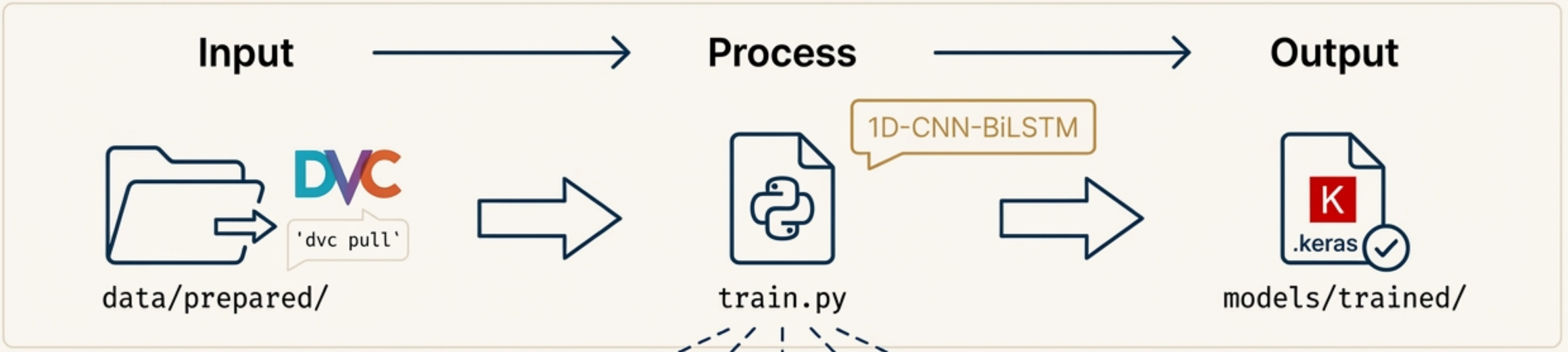

```
dvc add data/raw
```

DVC takes over tracking for the large XLSX files. Git now only stores a tiny .dvc pointer file, keeping the repository lightweight.

Stage 2: Transforming Raw Signals into ML-Ready Tensors



Stage 3: Training the Model with Full Experimental Visibility



Tool Spotlight: MLflow

mlflow

```
# Log parameters
run.log_params({"learning_rate": 0.001, "epochs": 50})

# Log final metrics
run.log_metrics({"accuracy": 0.95, "f1_macro": 0.93})

# Log artifacts
run.log_confusion_matrix(...)
run.log_keras_model(model,
                    registered_model_name="har-1dcnn-bilstm")
```

The Experiment Hub: Comparing Runs and Promoting Models

Run Comparison

Run Name	learning_rate	window_size	accuracy	f1_macro
training_v1	**0.001**	**200**	**0.95**	**0.93**
training_v2_lr_halved	0.0005	200	0.92	0.90

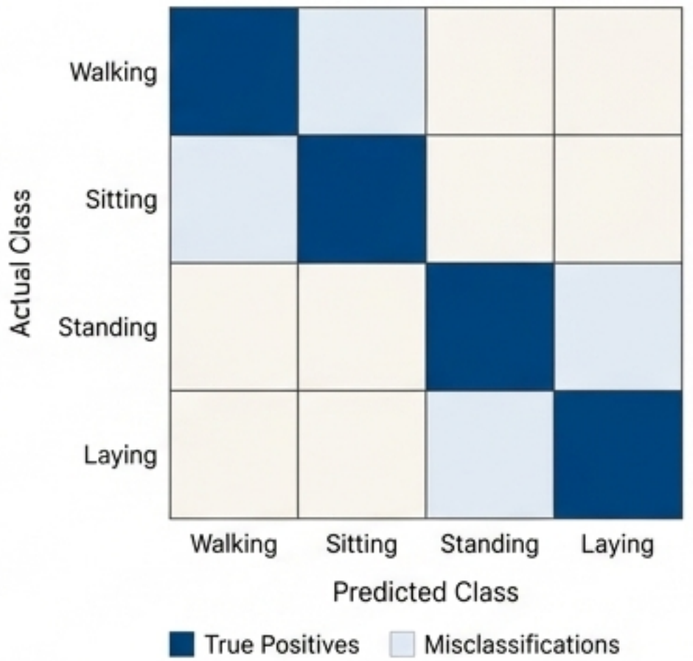
Model Registry

har-1dcnn-bilstm v1

Register Model

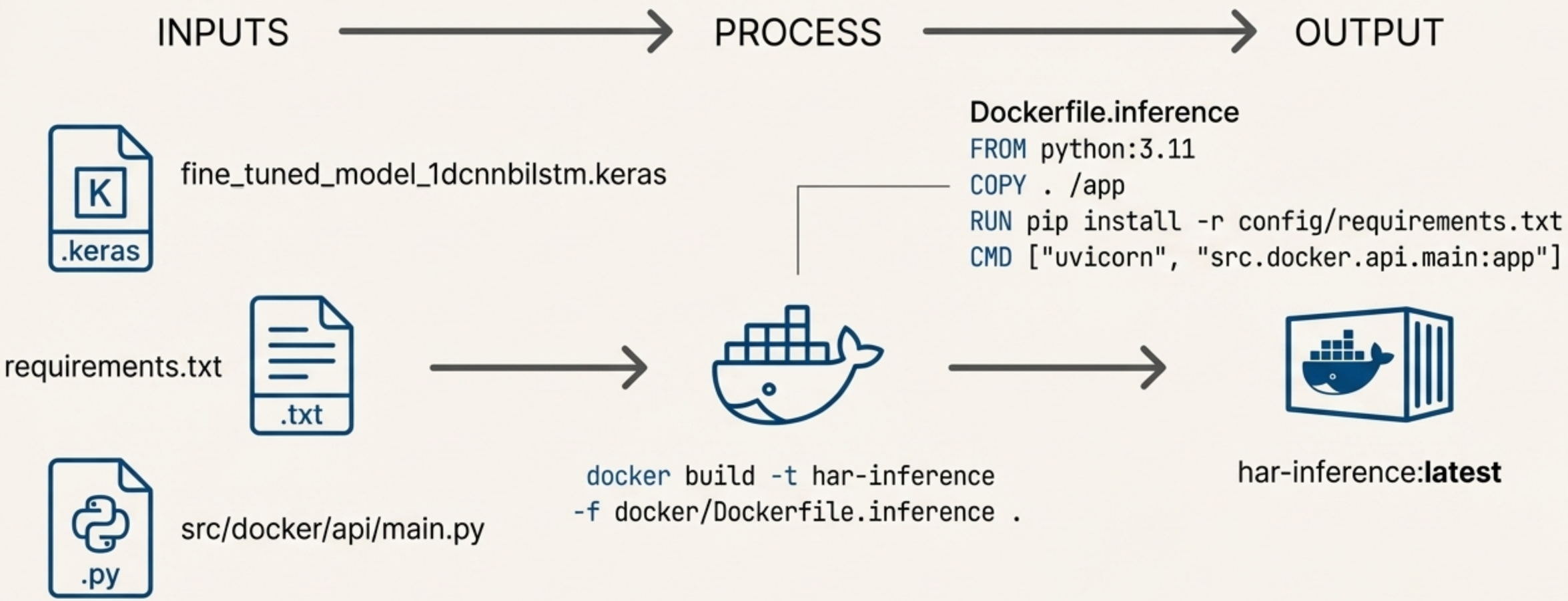
The best performing model is promoted to the Model Registry for staging and production.

Artifact Visualization

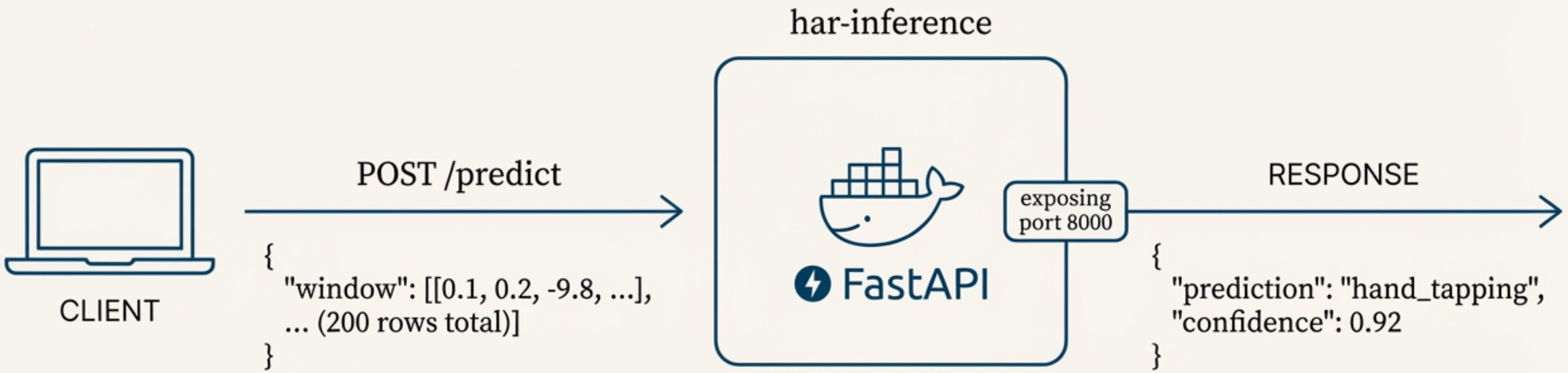


Accessible locally at http://localhost:5000 via Docker Compose.

Stage 4, Part 1: Packaging the Predictor for Production



Stage 4, Part 2: Serving Live Predictions on Demand



Tool Spotlight: Docker Compose

`docker-compose.yml`

The `docker-compose.yml` file orchestrates the entire service stack, including the `inference` API and the `mlflow` server.

A Key Finding: Solving the Domain Shift Catastrophe

Initial deployment resulted in the model predicting "hand_tapping" for 95% of all production data, despite high validation accuracy.

Root Cause Analysis

Training Data

Gravity component was **removed**. Mean Az acceleration: `~0 m/s²`.

Production Data

Gravity component was **present**. Mean Az acceleration: `-9.83 m/s²`.

Implemented a configurable Butterworth high-pass filter (0.3 Hz) in the preprocessing pipeline (`sensor_data_pipeline.py`) to remove the gravity vector from all incoming data, controlled via `config/pipeline_config.yaml`.

Before vs. After

Metric	Before Fix	After Fix
hand_tapping %	95.4%	4.2%
Unique Classes Predicted	4 / 11	7 / 11

The Workflow in Action: Onboarding New Participant Data

1. 
Place Raw Data
`cp new_participant.xlsx data/raw/`

2. 

Run Preprocessing
`python src/sensor_data_pipeline.py`
`python src/preprocess_data.py`

3. 
Version with DVC
`dvc add data`
`dvc push`

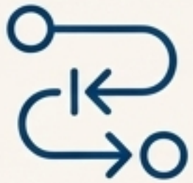
4. 
Commit Metadata to Git
`git commit -m "Add participant XXX"`

This standardized workflow ensures all new data is processed, validated, and versioned consistently before it is used for inference or retraining.

A Blueprint for Reproducible Machine Learning

Data & Model Versioning

Guarantees data integrity and full pipeline reproducibility. Every dataset and artifact is versioned and auditable.



Experiment Tracking & Governance

Provides complete experimental clarity. Every training run is logged, making results easy to compare, reproduce, and promote.

Environment Containerization & Deployment

Delivers production consistency. Every deployment is identical, eliminating environmental drift and the “it works on my machine” problem.

Explore the Project

GitHub Repository: The complete source code, documentation, and setup instructions are available on GitHub.

`ShalinVachheta017/MasterArbeit_MLops-`

API Documentation: Once running, interactive API documentation is available via the Swagger UI at `http://localhost:8000/docs`.