Composing XGBoost UDFs with Arrow Flight

Hussain Sultan
Xorq Labs
New York, United States
hussain@letsql.com

ABSTRACT

ML workflows suffer from fragmentation across SQL engines, Python feature engineering, and ML inference services. We present a unified approach using Arrow Flight RPC to bridge these systems through *User-Defined Exchange Functions* (UDXFs). Our framework enables seamless composition of ML operations while maintaining lineage and enabling cross-system optimizations. We demonstrate 5–20% performance improvements on XGBoost inference through automatic predicate push-down optimizations that prune unreachable decision tree paths.

VLDB Workshop Reference Format:
Hussain Sultan. Composing XGBoost UDFs with Arrow Flight. VLDB 2025 Workshop: Third International Workshop on Composable Data Management Systems.

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts are available at https://github.com/xorq-labs/xorq.

1 PROBLEM: ML WORKFLOW FRAGMENTATION

ML pipelines typically span SQL engines, Python scripts, ML frameworks, and REST APIs. Each transition breaks lineage tracking and forces custom integration code, making pipelines brittle and hard to optimize.

2 SOLUTION: FLIGHT-BASED UDXFS

We address this fragmentation through three integrated techniques that compose into a unified ML execution framework:

2.1 Arrow Flight as a Universal Interface

Instead of engine-specific UDFs, we package ML operations as *User-Defined Exchange Functions* (UDXFs) served over Arrow Flight RPC [1]. This treats SQL engines as RecordBatch transformers—a fundamental architectural shift requiring custom SafeTee and RecordBatchReader interfaces to enable truly streaming execution. Our implementation solves RecordBatch stream exhaustion issues that cause incorrect results in systems like DuckDB.

2.2 Deferred Execution with Lineage Preservation

Rather than immediate execution, we build a deferred relational graph using Ibis [2] expressions. This captures the complete ML workflow as a composable computation graph, enabling optimizations across system boundaries while preserving full lineage for replicability.

2.3 Cross-System Query Optimization

The deferred execution model enables novel optimizations. As a simple demonstration of using top-level query information for bespoke optimization, we show predicate push-down into XGBoost models: SQL filters are automatically extracted and used to prune unreachable decision tree paths via our experimental Quickgrove library [3]. While this approach yields measurable performance improvements, it remains opaque and algorithm-specific. More principled approaches like declarative sub-operators [5] offer a superior path forward, decomposing complex operators into declarative primitives that enable systematic optimization—a key direction for our future work.

3 DEMONSTRATION: END-TO-END MORTGAGE SCORING

Our live demonstration uses Fannie Mae mortgage data [4] with a pre-trained XGBoost model served as a Flight UDXF. We apply filters like `WHERE credit_score > 700` and show how the system rewrites the XGBoost UDXF to skip tree branches that cannot affect high-credit-score loans. Attendees will see execution plan differences and measured performance improvements in real-time, receiving complete source code and deployment guides.

REFERENCES

[1] *Arrow Flight: A High-Performance Protocol for Data Services.* arrow.apache.org/docs/format/Flight.html (2023).
[2] *Ibis: Python data analysis framework for Hadoop and SQL engines.* github.com/ibis-project/ibis (2024).
[3] *Quickgrove: Fast Tree Inference Library.* github.com/xorq-labs/quickgrove (2024).
[4] *Fannie Mae Single-Family Loan Performance Data.* capitalmarkets.fanniemae.com/credit-risk-transfer (2023).
[5] Jungmair, M. and Giceva, J., *Declarative Sub-Operators for Universal Data Processing.* Proceedings of the VLDB Endowment, 16(11), 3461-3474 (2023).
[6] Raven *et al.*, *End-to-End Optimization of ML Prediction Queries.* arXiv:2206.00136 (2022).
[7] *Xorq: Composable Data Management Framework.* github.com/xorq-labs/xorq (2024).
[8] *UDF Rewriting with Predicate Push-downs.* ibis-project.org/posts/udf-rewriting/ (2024).

ISSN 2150-8097.