SCALABLE LAKEHOUSE-DRIVEN ETL PIPELINE FOR REAL-TIME TICK DATA PROCESSING AND RELIABLE ML-POWERED TRADING DEPLOYMENT

[1] Ms. J. P. Aswini, Assistant Professor
Computer Science and Engineering, St. Joseph College of Engineering, Chennai-602117, Tamil Nadu
Email Id – aswinianchu1997@gmail.com

[2] Mr. K. Prasath, Student
Computer Science and Engineering, St. Joseph College of Engineering, Chennai-602117, Tamil Nadu
Email Id – saharaprasath@gmail.com

Abstract - **High-frequency trading requires scalable data infrastructure and production-grade machine learning pipelines. This project presents a full-stack Big Data Tick Data ML Trading architecture built for distributed processing and real-time deployment. Sub-second tick data is ingested, versioned, and stored using a lakehouse architecture based on S3, Apache Iceberg, and Postgres metadata. Apache Spark performs large-scale data cleaning, transformation, and feature engineering to generate ML-ready gold datasets. The machine learning lifecycle includes training, validation, testing, and walk-forward evaluation to ensure robustness. MLflow tracks experiments and model lineage, while FastAPI exposes validated models as live inference endpoints. Integration with MetaTrader 5 enables automated demo and live trade execution, bridging quantitative research with production trading systems.**

I. Introduction

In modern quantitative finance, high-frequency trading requires scalable and production-grade machine learning architectures. Traditional trading systems often lack proper data versioning, distributed processing, and structured deployment pipelines. This project presents a full-stack Big Data Tick Data ML Trading Pipeline designed for research-to-production continuity. The system ingests sub-second tick data and processes it using distributed frameworks like Spark. A Lakehouse architecture built on S3, Apache Iceberg, and Postgres ensures ACID guarantees and data version control.

Structured datasets are transformed into ML-ready gold tables for robust model training. The pipeline integrates model lifecycle management including training, validation, testing, and walk-forward analysis. MLflow ensures experiment tracking and reproducibility across model iterations. FastAPI exposes validated models as live endpoints for real-time signal generation. Finally, seamless MT5 integration enables automated demo and live trade execution in a scalable production environment.

II. Background and Motivation

A. Overview

This project implements a full-stack quantitative machine learning trading architecture designed for scalability, reproducibility, and production reliability. It ingests high-frequency tick data from Dukascopy at sub-second granularity and preserves raw CSV files as an immutable recovery layer to ensure auditability. Apache NiFi manages reliable large-scale data transfer into S3 object storage, while a Python processing layer converts raw CSV files into partitioned Parquet datasets for optimized storage and querying. Apache Spark performs distributed cleaning, sorting, and feature engineering to prepare ML-ready datasets. Data Version Control (DVC) ensures dataset lineage tracking and reproducibility across experiments.

The system is built on a Lakehouse architecture using S3, Apache Iceberg, and Postgres to provide ACID guarantees, schema evolution, and snapshot-based data management. Trino serves as the distributed SQL query engine for research validation and analytics. Gold datasets are structured for machine learning consumption, and the ML lifecycle includes training, validation, testing, and walk-forward evaluation to ensure robustness. MLflow tracks experiments, hyperparameters, and performance metrics, while validated models are deployed through a FastAPI live endpoint for real-time inference. Finally, integration with MetaTrader 5 (MT5) enables automated demo and live trade execution, effectively bridging quantitative research with scalable production trading deployment

B. Importance of the project

High-frequency trading demands robust, scalable, and fault-tolerant data infrastructure capable of handling massive volumes of sub-second tick data. Traditional trading systems often struggle with performance bottlenecks, lack proper data versioning, and fail to scale efficiently. This project ensures structured ingestion and processing of high-frequency datasets using distributed technologies. The lakehouse architecture provides ACID guarantees, schema evolution, and reliable data version control, ensuring consistency and integrity of financial data.

The use of Apache Spark enables large-scale distributed feature engineering and data transformation. Partitioned Parquet storage improves query performance and supports scalable analytics workflows. Iceberg tables introduce time-travel capabilities and controlled dataset evolution, while Trino enables distributed SQL analytics for research validation. Data reproducibility is maintained through DVC and MLflow experiment tracking, ensuring full transparency across model development cycles.

The structured ML lifecycle — including training, validation, testing, and walk-forward evaluation — reduces overfitting and improves robustness against market regime changes.

FastAPI deployment enables low-latency real-time inference, while MT5 integration bridges quantitative research models with live and demo trading environments. Separation of concerns between data, ML, and execution layers enhances maintainability and modular upgrades. Overall, the project establishes a production-grade, reproducible, and scalable ML trading ecosystem suitable for institutional-level deployment.

C. Motivation for This Research

High-frequency tick data is extremely large, complex, and generated at sub-second intervals, making it difficult to manage using traditional database systems. Conventional trading architectures often lack scalability and fail to process such massive datasets efficiently. As financial markets become increasingly data-driven, there is a growing demand for robust infrastructure capable of handling distributed, high-volume data processing.

Another major challenge in quantitative trading is the lack of reproducibility and proper data version control. Many trading systems do not preserve raw datasets properly, which affects auditability and reprocessing. Raw data preservation is essential to ensure transparency, traceability, and the ability to reconstruct experiments reliably. Without structured dataset lineage tracking, consistent research outcomes cannot be guaranteed.

Machine learning trading models frequently fail in production due to poor validation practices and overfitting. Regime shifts and changing market conditions require walk-forward validation strategies instead of static backtesting. Structured training, validation, and testing pipelines are necessary to ensure model robustness. Continuous experimentation also demands proper experiment tracking and comparable model versioning.

Scalable storage solutions such as lakehouse architectures provide improved reliability through ACID guarantees and schema evolution. Distributed processing frameworks are essential for handling sub-second financial data efficiently. Separation of data, ML, and execution layers enhances maintainability, modular upgrades, and system stability. This architectural separation ensures that improvements in one layer do not disrupt others.

Real-time inference APIs are crucial for live trading execution environments. Integration with platforms like MT5 bridges research models with actual trading systems. Automated deployment reduces operational risk and minimizes human error in financial execution. Ultimately, this research is motivated by the need to build a production-grade, scalable, and reproducible machine learning trading ecosystem that connects quantitative research seamlessly with live market deployment.**III. Novel Applications of the project**

The proposed system introduces a new dimension in **RegTech applications**, functioning as a real-time compliance monitoring platform for e-commerce. Regulatory authorities can automatically scan thousands of product listings to detect missing declarations such as MRP, net quantity, or manufacturer details. Instant alerts for non-compliant listings ensure faster enforcement, while batch URL scanning enables large-scale marketplace monitoring. This makes the system highly effective for both regulators and e-commerce companies, who can use it as a self-compliance verification tool before publishing products.

Beyond detection, the framework generates **structured audit reports** in PDF and CSV formats, supporting transparency and accountability. Compliance trend analysis helps identify frequent violators and high-risk product categories, while automated evidence collection strengthens legal enforcement proceedings. By integrating with government regulatory dashboards, the system provides centralized oversight, ensuring that compliance monitoring is both scalable and efficient.

The architecture also supports **cloud-based deployment**, enabling nationwide scalability and centralized monitoring. It can be extended to cross-border trade compliance verification, enhancing transparency in global product declaration standards. Real-time logging ensures audit trail maintenance, while dynamic rule updates allow the system to adapt as legal standards evolve. This flexibility ensures long-term relevance in a rapidly changing regulatory environment.

Overall, the project establishes an **intelligent and scalable digital ecosystem** for Legal Metrology compliance enforcement. By reducing manual workload, operational costs, and inspection delays, it enhances consumer protection and regulatory efficiency. With potential AI integration for predictive compliance risk analysis, the system represents a forward-looking solution that bridges regulatory requirements with the realities of modern e-commerce.

IV. Role and Potential

1. Role in Strengthening Regulatory Enforcement

The system plays a critical role in managing high-frequency tick data efficiently. It enables distributed data processing using Apache Spark for large-scale computations. Lakehouse architecture ensures structured and reliable financial data storage. Partitioned Parquet datasets improve performance for analytical queries. Data versioning ensures reproducibility and auditability in research workflows. Thus, it strengthens scalable and production-grade financial data engineering practices

2. Role in Consumer Protection and Transparency

The framework enforces structured training, validation, and testing phases. Walk-forward validation reduces overfitting and improves regime robustness. MLflow tracks experiments, hyperparameters, and model performance metrics. Dataset lineage is preserved using DVC for controlled experimentation. Model evaluation ensures only validated strategies reach deployment. Overall, it enhances reliability and accountability in ML-based trading systems.

3. Potential for Scalable and Intelligent Monitoring

FastAPI enables low-latency live model inference for trading signals. Integration with MT5 bridges research models to live and demo markets. The separation of model logic and execution logic improves maintainability. The architecture supports multi-asset and multi-strategy deployment. Cloud scalability allows high-throughput production environments. This demonstrates strong potential for institutional-grade automated trading systems.

4. Potential for Cross-Sector and Enterprise Integration

The modular architecture allows independent upgrades of data and ML layers. Time-travel capabilities support historical reconstruction and deep backtesting. Researchers can experiment without disrupting production workflows. The system enables continuous retraining with controlled dataset evolution. It supports expansion into alternative data and advanced model architectures. Thus, the project establishes a future-ready and innovation-driven quantitative trading ecosystem.

V. Conclusion

This project successfully presents a full-stack, production-oriented quantitative machine learning trading architecture designed for scalability, reliability, and reproducibility. By integrating distributed data processing, lakehouse storage, strict data version control, and structured ML lifecycle management, the system addresses the major challenges of handling high-frequency tick data. The architecture ensures auditability through raw data preservation and snapshot-based versioning while enabling efficient feature engineering using Spark. Walk-forward validation and structured training phases enhance robustness against market regime changes. MLflow tracking ensures experiment transparency and controlled model evolution. Real-time FastAPI deployment combined with MT5 integration bridges the gap between research and live trading environments. The modular separation of data, ML, and execution layers improves maintainability and extensibility. Overall, the project establishes a scalable, research-driven, and production-ready ML trading ecosystem capable of continuous innovation and institutional-grade deployment.

VI. Future Research Directions

Integrating LSTM, Transformer, and attention-based models for improved sequential tick data modeling.

- Reinforcement Learning for Trade Execution:
  Applying Deep Reinforcement Learning to optimize entry, exit, and position sizing strategies.
- Alternative Data Integration:
  Incorporating news sentiment, macroeconomic indicators, and order book data for enhanced signal generation.
- Real-Time Stream Processing:

Extending the pipeline with Kafka or Spark Streaming for true real-time ingestion and processing.

- Risk Management Layer Enhancement:
  Developing dynamic risk control modules including volatility-based position sizing and drawdown control.
- AutoML and Hyperparameter Optimization:
  Implementing automated model tuning frameworks for systematic performance improvement.
- Multi-Asset Portfolio Optimization:
  Expanding from single-symbol models to portfolio-level optimization and capital allocation strategies.
- Low-Latency Infrastructure Optimization:
  Improving inference latency using optimized deployment frameworks and container orchestration.
- Explainable AI (XAI) for Trading Models:
  Integrating SHAP or feature attribution techniques to interpret model decisions.
- Cloud-Native Distributed Deployment:
  Designing Kubernetes-based scalable deployment for institutional-grade production environments

.Reference:

[1] Mei Long, "Understanding ETL Modernization," Prophecy, Feb 3, 2025.

[2] Michael Segner, "Data Pipeline Architecture Explained: 6 Diagrams and Best Practices," Monte Carlo, March 31, 2023.

[3] Pradeep Kumar Vattumilli, "Metadata-Driven ETL Pipelines: A Framework for Scalable Data Integration Architecture," International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), 19 December 2024.

[4] Seth Rao, "Data Quality Management: Framework and Metrics for Successful DQM Model," FirstEigen, 28 November 2024.

[5] Data checks, "The State of Data Quality 2024: Analysis of 1000+ Data Pipelines," DataChecks, 6 November 2024.

[6] Raj, "Error Handling in Data Pipelines," Hey Coach Blogs, 27 December 2024.

[7] Robert Kelly, "Enterprise Delivery Pipeline as a Product," liatrio, 9 August 2017.

[8] Allison Foster, "What is Enterprise Data Processing? Key Concepts and Best Practices," Sqream, 24 December 2024.

[9] Jeffrey Richman, "What Is A Cloud Data Pipeline? Types, Benefits, & Use Cases," Estuary, 23 July 2023.

[10] Alex Solutions, "How Automated Data Lineage Powers Enterprise Data Governance & Compliance," Alex