Senior Design Project Proposal:

Cloud AutoML Toolchain leveraging Domain-Specific LLMs for Expert Workflows

Shree Chaturvedi

CSE Department, Miami University

CSE 448: Senior Design Project I

Prof. Lynn Stahr

August 30, 2025

Automated Data Scientist Platform

The platform is a cloud-hosted, AI-augmented MLOps toolchain for data scientists tackling high-dimensional, context-rich problems (e.g., multivariate time-series forecasting in satellite telemetry or graph-based protein interaction modeling in biotech). It orchestrates an AutoML workflow, from ingestion, EDA, feature engineering, training, to deployment and monitoring, using hybrid LLM-driven automation. We'll be using RAG for injecting unstructured domain knowledge into structured pipelines, while enforcing user-overridable transparency to mitigate black-box pitfalls.

Workflow Phases

Data Ingestion Engine: Supports multi-format structured inputs (CSV, JSON, XLSX, Parquet, SQL dumps) and unstructured context (PDFs, MD, DOCX, research corpora). Backend pipeline: schema inference via Pandas/SQL Alchemy; data quality profiling (nulls, outliers, skew via Great Expectations); inter-dataset joins via semantic matching. Unstructured docs are parsed and embedded into vector DB (pgvector) for RAG. Optional SFT on a small base LLM (e.g., Llama-3.1-8B, Phi-3-mini via PEFT/QLoRA) using context as training corpus to adapt for domain jargon (e.g., "thermal stress thresholds" in aerospace). LLM then proposes prediction targets via RAG-prompted generation (e.g., "regress satellite longevity from orbital decay" retrieved from NASA specs), output as structured JSON for UI injection.

Exploratory Data Analysis (EDA) & Querying: User types a query in NLQ which is translated to SQL via fine-tuned LLM (post-SFT, chained with RAG for context-specific query gen, e.g., "aggregate sensor variances during anomaly windows per doc-defined criteria"); direct SQL fallback for precision. Executes on in-memory cache (Redis) or columnar store (DuckDB) for low latency on large datasets. Outputs: interactive visualizations (distributions, correlations, PCA projections) with LLM insights (e.g., "outlier cluster aligns with protein misfolding motifs from literature").

Domain-Aware Feature Engineering: Hybrid generator provides statistical baselines (rolling stats, polynomial interactions, FFT transforms via SciPy) augmented by RAG-retrieved suggestions (LLM queries vector DB for passages, e.g., "derive Gibbs free energy delta from sequence data per biotech reports"). Outputs JSON schema: {feature_name, description, method (e.g., "linear_regression_slope"), params (e.g., {"window": 7})}. UI as dynamic control panel: rows with toggles for inclusion, editable inputs for params (e.g., slider for percentile thresholds), previews of impact (SHAP values or partial dependence plots).

Model Training & Optimization: Flexible typology: classification/regression/forecasting via templates (XGBoost, LightGBM, Prophet; hyperparams via Optuna/Bayesian opt). Preprocessing code LLM-generated (RAG-informed, e.g., "impute via domain-specific interpolation from telemetry guidelines") but user-guidable. LLM uses MCP tools to generate training code, runs code in provided environment, and iterates until ML/model problem is solved based on problem-context and user query. User is shown code cell summaries and has option to view actual code. At any step, the user can stop the LLM and steer it in a different direction or interact with the Jupiter notebook themselves.

Deployment, Monitoring, & Interpretation: Auto-containerizes models (Docker) for API deployment (FastAPI endpoints, scalable via Kubernetes). Monitoring dashboard: real-time metrics (latency, throughput), drift detection (KS-test on features), alerts, etc. Interpretability: LLM-postprocessed explanations (e.g., "prediction driven by decay rate feature, per doc-cited physics model") using LIME/SHAP outputs grounded in RAG. Lifecycle: model registry with versioning, A/B testing hooks, rollback.

Differentiating Technical Innovations

RAG-Centric Domain Injection: Vector embeddings (SentenceTransformers) enable retrieval-augmented prompting, outperforming zero-shot LLMs by 20-30% in feature relevance per recent benchmarks; mitigates hallucinations via top-k reranking.

Structured LLM Outputs for Determinism: All LLM gens formatted as JSON schemas (via function-calling in LangChain), parsed to build reactive UI (React components), hiding non-determinism. Exposes params for tweaking by a data scientist or expert user. This avoids "debug generated code" anti-pattern.

Hybrid Automation Spectrum: Conceptual "express lane" for end-to-end auto (ingest-to-deploy in minutes via defaults); "granular control" at every step;

MLOps Backbone: Project isolation ensures data sovereignty; cloud arch (S3 storage, Lambda compute, pgvector DB) for elasticity; CI/CD for iterations; bias audits (e.g. AIF360) integrated into dashboards.

Feasibility & Value

Total developer time estimate is ~1200 hours: MVP prioritizes ingestion-EDA-features (core RAG value); stretch to training/deployment. Leverage existing low-level libraries to focus on building functionality. Demo: end-to-end AutoML on public datasets (e.g., NASA CMAaPSS telemetry).