Husayn El Sharif

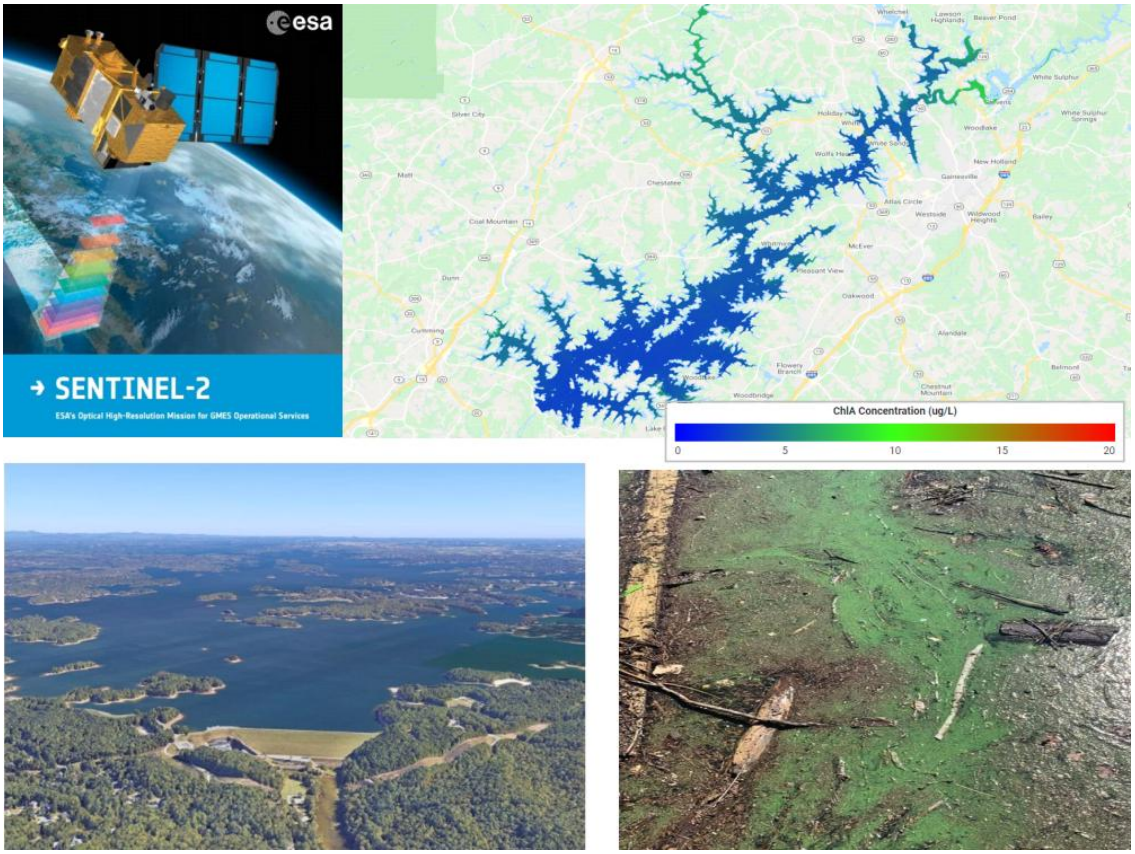
Senior Data Scientist • Machine Learning Engineer

Marietta, GA helsharif@gmail.com (561) 247-1430 hesportfolio.com husayn-el-sharif

Portfolio

TABLE OF CONTENTS

Remote Sensing & ML for Water Quality Assessment at Lake Lanier



Remote Sensing & ML for Water Quality Assessment at Lake Lanier

Remote Sensing • Symbolic Regression • Time-Series Analysis • Google Earth Engine • Dockerized Deployment • Regulatory Compliance

Quick Facts

Role	Lead Data Scientist / ML Engineer
Domain	Water Quality, Remote Sensing, Early-Warning Systems
Tech Stack	Python, Symbolic Regression, Sentinel-2, Google Earth Engine, JupyterLab, Docker
Stakeholder	Gwinnett County Dept. of Water Resources

This project developed a satellite-based workflow for estimating chlorophyll-a (Chl-a) in Lake Lanier, Georgia, using Sentinel-2 imagery and in-situ sampling. Chl-a is a key indicator of algal biomass and eutrophication, and is central to monitoring harmful algal blooms and water quality standards compliance.

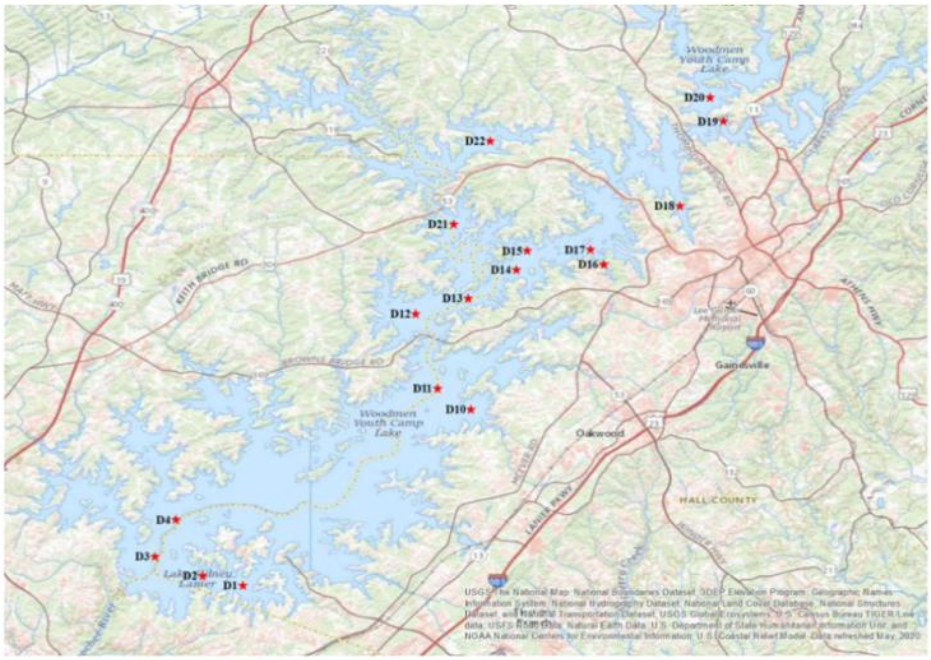
Traditional field sampling is expensive and sparsely distributed in space and time. By mining satellite reflectance data and learning robust relationships to in-situ measurements, this workflow produces weekly, lake-wide Chl-a maps that give water utilities and regulators a much richer view of changing conditions.

Background & Problem Statement

Chlorophyll-a is widely used as a proxy for algal biomass and trophic state in lakes and reservoirs. Elevated Chl-a levels can indicate harmful algal blooms that threaten ecosystem health, drinking water safety, and recreation.

In Lake Lanier, routine field assessments provide high-quality measurements but are limited in spatial and temporal coverage and require significant field and lab effort. This creates blind spots in understanding how water quality evolves across the full lake surface, especially between sampling events.

Problem Statement: How can satellite-based remote sensing using Sentinel-2 imagery be used to assess and monitor chlorophyll-a concentrations in Lake Lanier, GA, in near real time to improve water quality management and detect harmful algal blooms?



Lake Lanier Chl-A Field Sampling Locations

Band	Spectral region	Wavelength range (nm)	Resolution (m)
B1			
B2	Blue	458–523	10
B3	Green peak	543–578	10
B4	Red	650–680	10
B5	Red edge	698–713	20
B6	Red edge	733–748	20
B7	Red edge	773–793	20
B8	NIR	785–899	10
B8A	NIR narrow	855–875	20
B11	SWIR	1565–1655	20
B12	SWIR	2100–2280	20

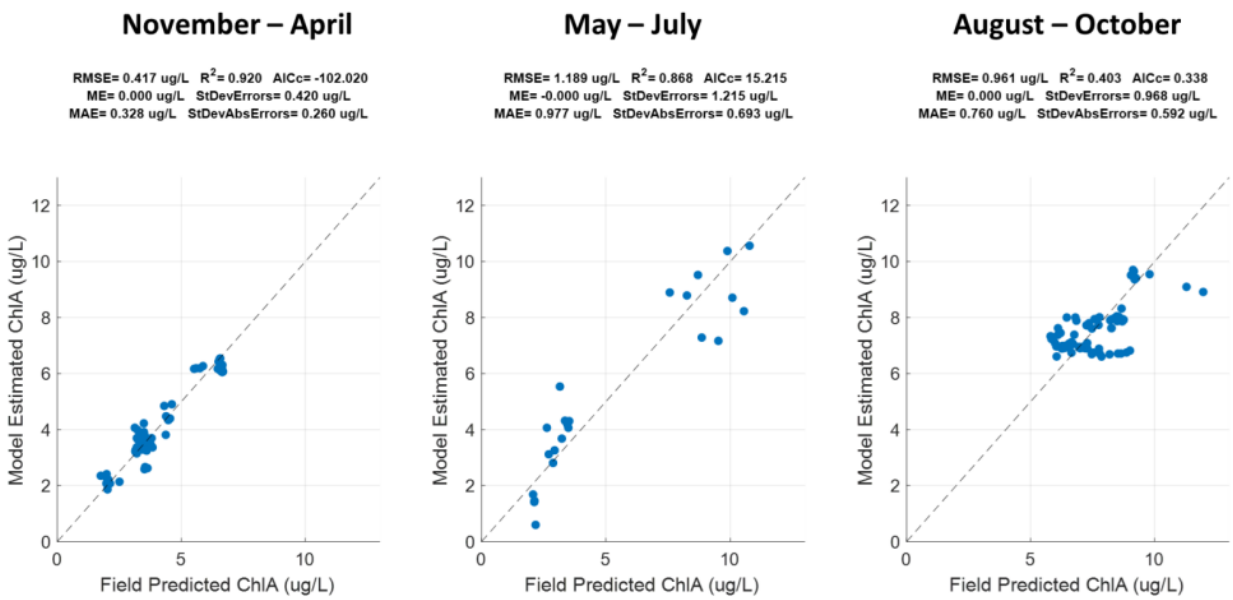
Sentinel-2 Reflectance Bands

Lake Lanier Chl-A Field Sampling Locations and Sentinel-2 Reflectance Bands

Model Development

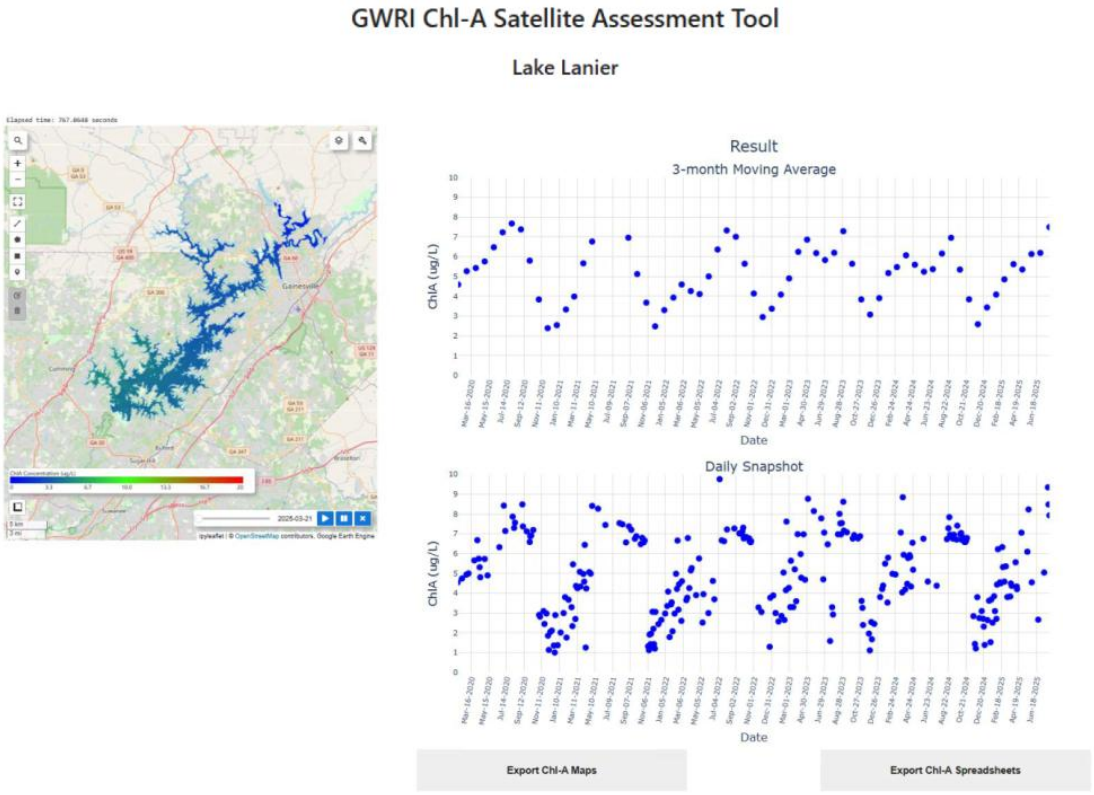
A multi-year lake sampling campaign was conducted from fall 2018 through 2020. During Sentinel-2 overpasses, photic-zone chlorophyll-a concentrations were measured at multiple locations across Lake Lanier. The resulting paired dataset of in-situ Chl-a and satellite reflectance formed the foundation for model development.

- Assembled a dataset linking in-situ Chl-a measurements with Sentinel-2 surface reflectance bands for each sampling location and overpass date.
- Segmented the data by season to account for changing optical properties and stratification in the lake.
- Applied Symbolic Regression to discover functional relationships between Sentinel-2 bands and Chl-a, identifying the most informative spectral combinations for water quality prediction.
- Evaluated alternative functional forms and seasonal models to balance predictive accuracy, robustness, and interpretability for operational use.



Regression Analysis: Estimating Chl-A (target variable) as functions of Sentinel-2 Reflectance Bands

Regression Results for Chlorophyll-a Estimation from Sentinel-2 Bands



Web-based Interactive Dashboard for Lake Lanier Water Quality Assessment

Chlorophyll-a Estimation Tool Dashboard for Lake Lanier

Model Deployment & Results

Once robust Chl-a models were established, they were deployed on Google Earth Engine (GEE) , enabling near real-time processing of Sentinel-2 scenes and generation of operational water-quality products.

- Implemented the Chl-a estimation algorithms as GEE scripts to automatically ingest new Sentinel-2 imagery, apply atmospheric corrections, and compute lake-wide Chl-a maps.
- Built interactive browser-based tools that allow users to visualize spatial patterns, query locations, and export maps and data tables for further analysis.
- Developed a JupyterLab-based companion workflow for power users, supporting exploratory analysis, time-series extraction, and custom reporting.
- Containerized the full analysis environment with Docker so the tool can be launched consistently on different machines or deployed in cloud / enterprise environments without dependency issues.

Impact & Actionable Insights

The deployed Chl-a Sat-Tool now serves as a decision-support platform for the Gwinnett County Department of Water Resources , providing a synoptic, near real-time view of water quality conditions across Lake Lanier.

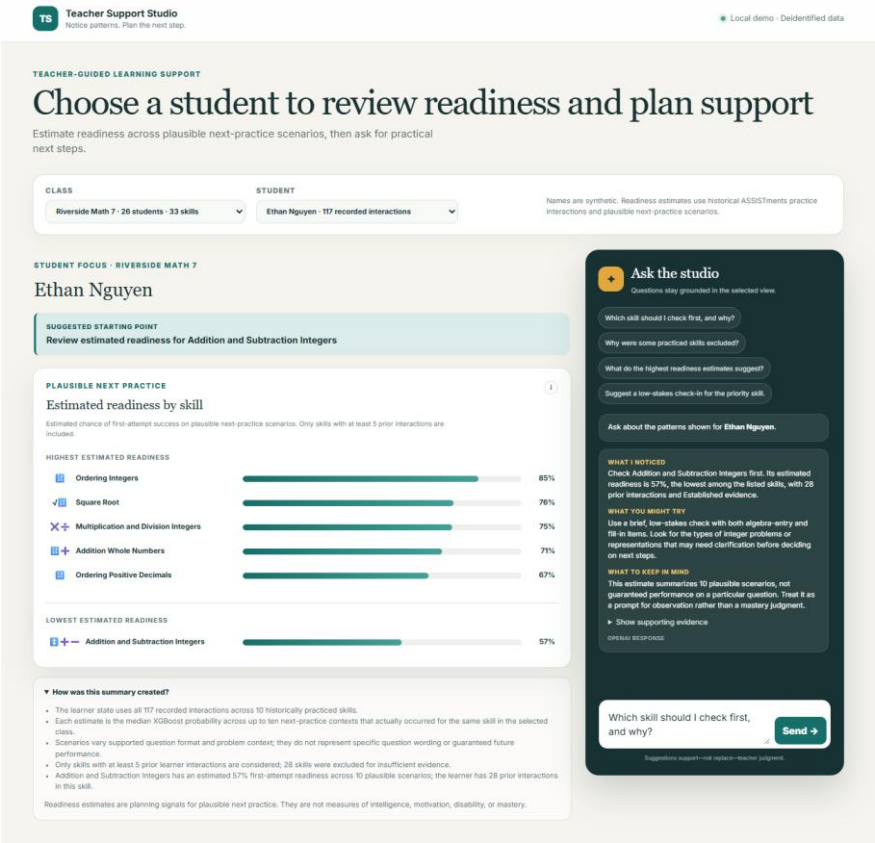
- Enables long-term lake monitoring at a fraction of the cost of traditional field campaigns and laboratory analysis.
- Reveals that large areas of Lake Lanier frequently exceed regulatory thresholds for chlorophyll-a, supplying data-driven evidence to prioritize remediation and regulatory compliance efforts.
- Supports early detection of harmful algal blooms by highlighting hotspots and emerging trends between routine sampling events.
- Demonstrates a reproducible, containerized workflow that can be adapted to other reservoirs and regions, accelerating adoption of satellite-based water quality monitoring.

Conference Presentation

I presented this project at the 2021 Georgia Water Resources Conference in Athens, GA, held March 22-23, 2021.

https://www.youtube.com/embed/IcQ7qOMTyN0

Teacher Support Studio: Student Proficiency Prediction & AI Teacher Assistant



Teacher Support Studio end-to-end educational machine learning workflow

Educational Data Science • Longitudinal ML • XGBoost • Logistic Regression • SHAP • FastAPI • LangGraph • OpenAI • Responsible AI

Quick Facts

Role	Data Scientist / Machine Learning Engineer / Full-Stack AI Developer
Dataset	ASSISTments 2009-2010 Skill Builder: 259,386 filtered interactions from 4,163 students
Objective	Estimate the probability of first-attempt success on a student's next skill-practice interaction
Best Model	Tuned XGBoost with 0.70 test ROC-AUC and 0.85 average precision
Application	FastAPI readiness dashboard with SHAP explanations and a LangGraph-powered teacher assistant
Live Demo	Open Teacher Support Studio

Teacher Support Studio is an end-to-end educational machine learning project that predicts a student's probability of first-attempt success on upcoming skill practice and turns those estimates into teacher-facing, evidence-grounded guidance.

The project combines leakage-safe longitudinal feature engineering, interpretable and nonlinear classification models, model-native SHAP contributions, and a deployable FastAPI application. Its outputs are designed as low-stakes planning signals - not mastery labels or automated instructional decisions.

Background & Problem Statement

Student practice histories contain useful signals about recent performance, prior learning opportunities, help seeking, and skill-specific progress. Used carefully, those signals can help an educator decide where a brief check-in or additional practice may be useful.

Machine Learning Question: What is the probability that a student answers the next skill-practice interaction correctly on the first attempt?

Application Question: How can that probability be presented with its supporting evidence and limitations so teachers retain contextual interpretation and final decision-making?

Dataset & Leakage-Safe Feature Engineering

The modeling workflow uses the ASSISTments 2009-2010 Skill Builder dataset. After filtering to original problems with valid skill identifiers, the working data contains 259,386 interactions from 4,163 students and an overall first-attempt correctness rate of 65.8%.

- 170 documented predictors: student, skill, practice, recency, streak, opportunity, and help-seeking signals.
- Past-only information: longitudinal features are shifted so each row uses only evidence available before the predicted interaction.
- Chronological evaluation: unique interaction order values are split into the earliest 70% for training, the next 15% for validation, and the latest 15% for testing.
- Fair comparison: logistic regression and XGBoost use the same predictor set and held-out evaluation protocol.
- Protected test window: operating thresholds are selected on validation data rather than tuned against final test outcomes.

Modeling Workflow & Evaluation

An L2-regularized logistic regression provides an interpretable baseline, while tuned XGBoost captures nonlinear relationships and interactions in students' learning histories. Both models are evaluated on discrimination, ranking quality, probability error, and threshold-based balance.

Held-Out Test Performance

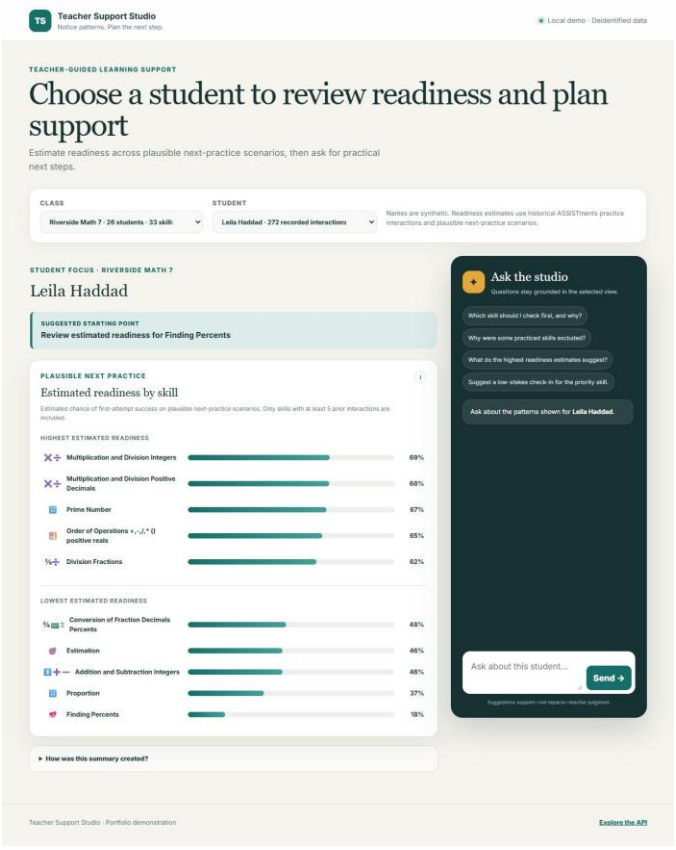
Model	ROC-AUC	Average Precision	Log Loss	Brier Score	Balanced Accuracy at Selected Threshold
Logistic regression	0.67	0.83	0.56	0.19	0.61 at 0.59
XGBoost	0.70	0.85	0.54	0.18	0.63 at 0.64
Prevalence baseline	0.50	0.72	0.61	0.21	0.50

- Consistent improvement: XGBoost outperforms logistic regression across discrimination and probability-error metrics.
- Strongest signal: recent student-skill performance - especially mean skill accuracy across the five latest relevant interactions - drives the nonlinear model.
- Interpretability: global feature importance and local SHAP contributions connect predictions to the learner evidence behind them.

- Important limitation: the final test window contains 38,909 interactions but only 87 students, so performance is not evidence of broad unseen-student generalization.

Teacher Support Studio Application

The application lets a teacher select a synthetic class and student, review estimated readiness across named skills, inspect how each summary was created, and ask questions grounded in the evidence currently displayed.



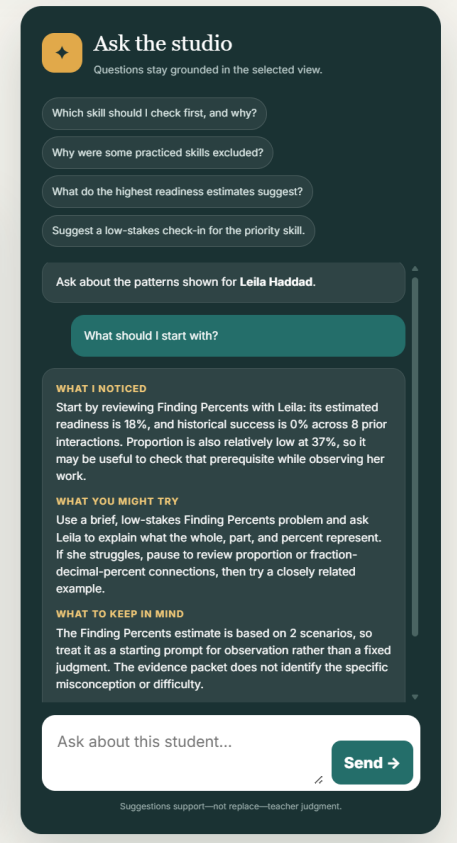
Teacher Support Studio readiness dashboard with synthetic student names and deidentified data

Readiness Scoring Workflow

1. The teacher selects a synthetic class and student mapped to authentic deidentified identifiers.
2. The application retrieves the student's complete recorded interaction history.
3. It constructs up to ten historically observed next-practice contexts for each named skill.
4. The persisted XGBoost model scores the scenarios, and their median probability becomes the skill's estimated readiness.
5. Skills with fewer than five prior learner interactions are excluded before the five highest and five lowest estimates are selected.
6. A LangGraph workflow gives OpenAI only the selected metrics and evidence for a structured response; deterministic local guidance remains available if OpenAI is unavailable.

Grounded AI Teacher Assistant

The assistant answers preloaded or free-form questions in three predictable sections: what the model suggests, a low-stakes action the teacher might try, and limitations to keep in mind. The prompt receives only the selected student's displayed metrics and supporting evidence, keeping the response tied to the current view.



Grounded teacher assistant with evidence-aware recommendations and responsible-use guidance

Application Architecture & Reproducibility

- Data and modeling: Python, pandas, NumPy, SciPy, scikit-learn, and XGBoost.
- Interpretability: XGBoost model-native SHAP contributions, Plotly, and Kaleido.
- Application layer: FastAPI, Uvicorn, Pydantic, JavaScript, HTML, and CSS.
- Generative AI: LangGraph, LangChain, and OpenAI with a deterministic fallback path.
- Quality and reproducibility: uv, JupyterLab, pytest, Ruff, persisted model artifacts, and ordered notebooks.
- Deployment: a self-contained Render bundle packages the application, model, mappings, and selected demo data.

Responsible Use & Limitations

This project is a portfolio demonstration of teacher-facing decision support. Predictions are planning signals for plausible next practice - not measures of intelligence, general ability, motivation, disability, or psychological mastery.

- Do not use estimates to automatically determine grading, placement, access, or interventions.
- Treat the five-interaction minimum as a demo evidence rule rather than a validated educational standard.
- Interpret results in light of temporal and population shift in the held-out data.
- Validate calibration, subgroup behavior, unseen-student generalization, and instructional impact before real educational use.
- Keep educators responsible for contextual interpretation and final decisions.

GitHub Repository & Live Demo

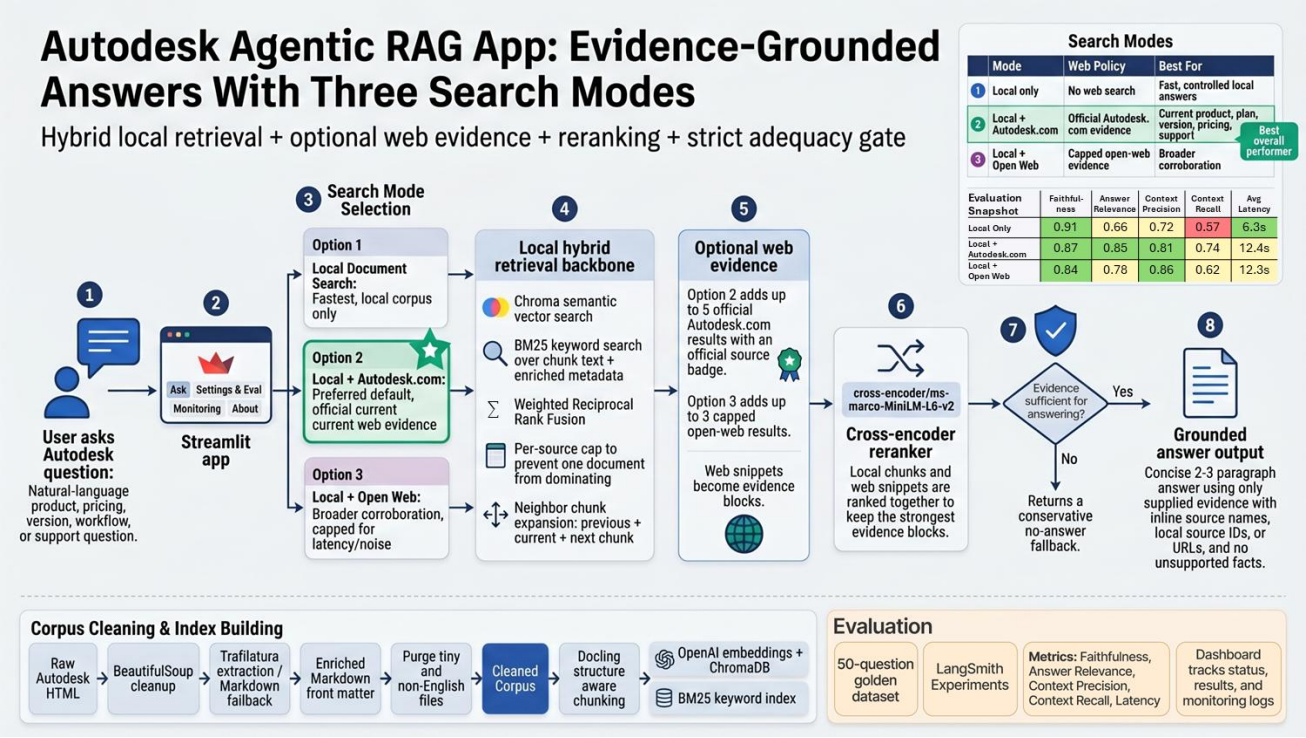
Explore the complete analysis, application source, reproducible environment, tests, deployment bundle, and interactive teacher-facing demo.

🔗 View Project Repository on GitHub

🚀 Open Live Demo

The live Render service may need a short warm-up after a period of inactivity.

Autodesk Agentic RAG Chatbot



Autodesk RAG app architecture and workflow overview

Agentic RAG • Hybrid Retrieval • Chroma • BM25 • Weighted RRF • Cross-Encoder Reranking • LangSmith • Streamlit • Hugging Face Spaces • Docker • Supabase Monitoring

Quick Facts

Role	Machine Learning Engineer / RAG Developer
Domain	Autodesk product knowledge, software support, CAD/CAM/CAE workflows, and customer-facing technical Q&A
Tech Stack	Python, Streamlit, LangChain/LangSmith, Chroma, BM25, OpenAI embeddings, cross-encoder reranking, SerpAPI, Supabase, Docker, Hugging Face Spaces
Corpus	1,218 raw Autodesk HTML pages cleaned to 974 retained Markdown documents and 19,611 indexed chunks
Search Modes	Local document search, local + Autodesk.com official web evidence, and local + capped open-web evidence
Recommended Demo Mode	Option 2: Local Document Search + Autodesk.com, based on the strongest overall evaluation balance
Repository	github.com/helsharif/autodesk-rag-app
Live Demo	huggingface.co/spaces/helsharif/autodesk-rag-assistant

This project is an agentic retrieval-augmented generation application for answering Autodesk product and workflow questions with grounded local evidence, optional web verification, source citations, and conservative no-answer behavior when evidence is insufficient.

The system moves beyond a basic chatbot by combining corpus cleaning, dense and lexical retrieval, reciprocal rank fusion, source caps, same-document context expansion, cross-encoder reranking, adequacy checks, prompt-injection hardening, LangSmith evaluation, monitoring hooks, and a public Docker deployment on Hugging Face Spaces.

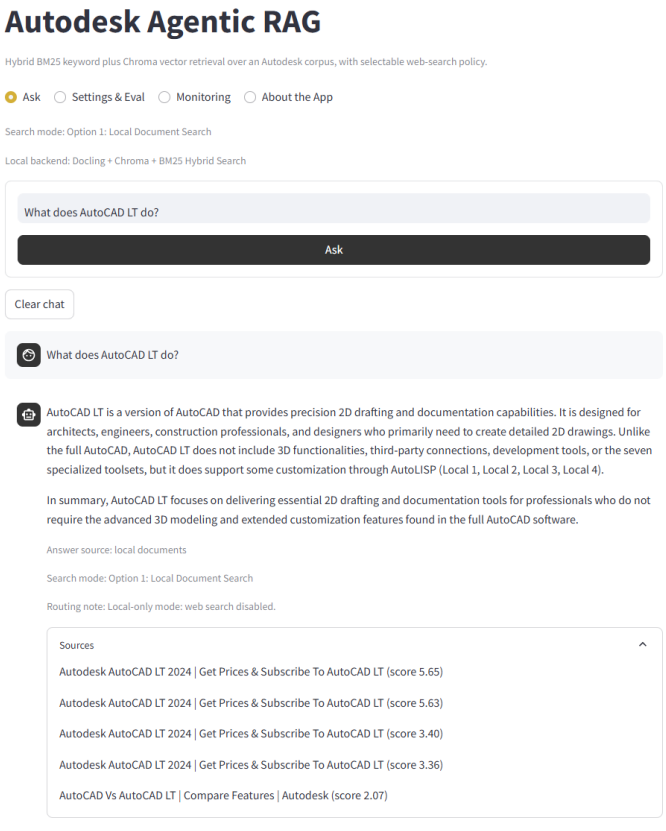
Background & Problem Statement

Autodesk product information spans product pages, support content, feature descriptions, system requirements, licensing pages, industry workflows, and comparison pages. A user may ask a natural-language question about Fusion, AutoCAD, Revit, Inventor, Maya, or related workflows, but the answer often requires retrieving the right product-specific evidence and avoiding unsupported generalizations.

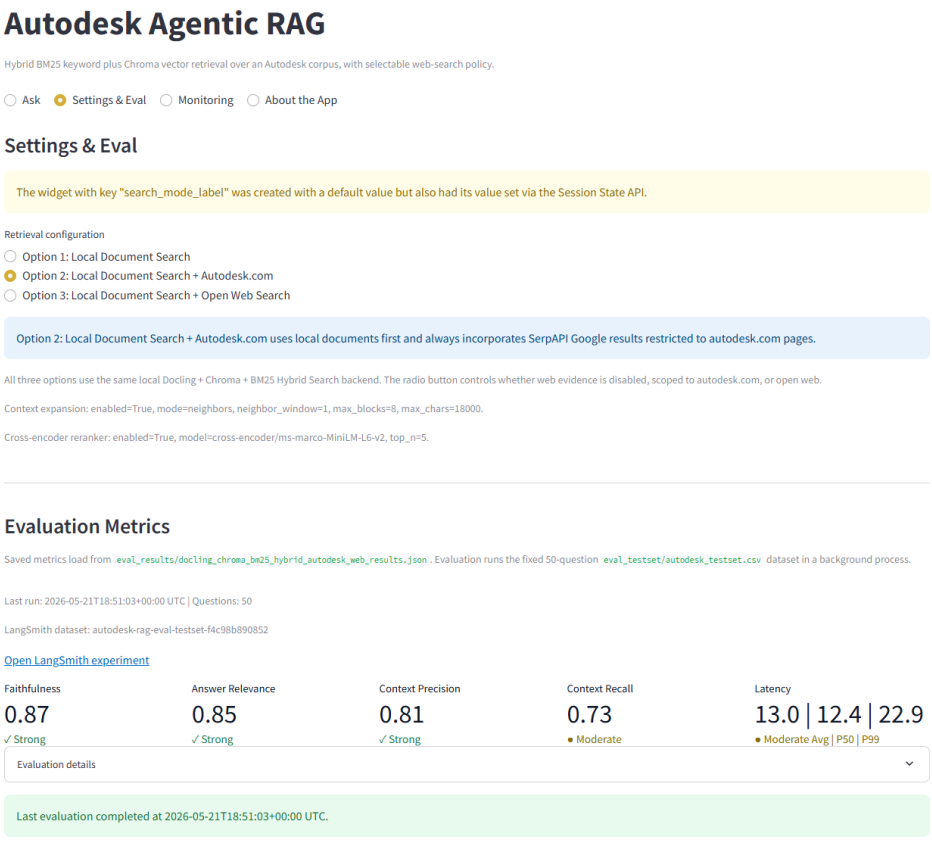
Problem Statement: Can a hosted RAG assistant answer Autodesk product and workflow questions with concise, cited responses while combining local corpus evidence, official Autodesk.com verification, open-web corroboration when selected, and strict safeguards against unsupported or prompt-injected answers?

Streamlit App Interface

The Streamlit interface is organized around four review-friendly areas: Ask, Settings & Eval, Monitoring, and About the App. Users can submit Autodesk questions, switch search modes, inspect cached evaluation metrics, and review runtime monitoring summaries when Supabase is configured.



Ask tab for the Autodesk RAG assistant



Settings and evaluation tab for selecting search mode and reviewing metrics

Live Demo Walkthrough

The demo video shows the hosted application flow, including asking Autodesk product questions, selecting evidence policies, and reviewing how the assistant presents sourced answers.

https://hesportfolio.com/assets/projects/project-autodesk-rag-app/autodesk-rag-demo.mp4

Corpus Cleaning & Indexing

The data pipeline converts raw Autodesk HTML pages into RAG-ready Markdown and retains metadata needed for source-grounded retrieval. The cleaning stage removes boilerplate, drops very small or non-English pages after enrichment, and writes per-file diagnostics for review.

- Raw HTML files found: 1,218 Autodesk pages.
- Retained cleaned documents: 974 Markdown files after purge rules.
- Raw-to-cleaned size reduction: 303.27 MB of HTML to 4.37 MB of Markdown.
- Character reduction: 317,446,072 raw characters to 3,574,578 cleaned characters, approximately 98.87% smaller.
- Indexed chunks: 19,611 chunks stored in both Chroma and BM25 artifacts.
- Embedding model: OpenAI text-embedding-3-small with 1,536-dimensional vectors.

Agentic RAG Architecture

The runtime agent treats the user question, retrieved local excerpts, and web snippets as untrusted inputs. It applies deterministic safety checks, routes the query, retrieves local and optional web evidence, reranks the combined candidates, checks answerability, and only then generates a concise answer with citations.

- Security screen: blocks obvious jailbreak, prompt-reveal, secret-exfiltration, and unsafe cyber requests before the router LLM.
- Query routing: identifies Autodesk relevance, current/latest/pricing needs, and compare/contrast questions.
- Hybrid retrieval: searches Chroma vectors and BM25 keyword indexes in parallel.
- Rank fusion: merges dense and lexical results with weighted reciprocal rank fusion and per-source caps.
- Compare-aware retrieval: extracts product entities, retrieves focused subqueries, deduplicates, and balances evidence across compared products.
- Context expansion: adds neighboring chunks from the same source document to reduce chunk-boundary failures.
- Reranking: uses cross-encoder/ms-marco-MiniLM-L6-v2 to rerank local and web evidence together.
- Adequacy gate: returns a conservative no-answer response when evidence does not explicitly support an answer.

Search Modes & Retrieval Engineering

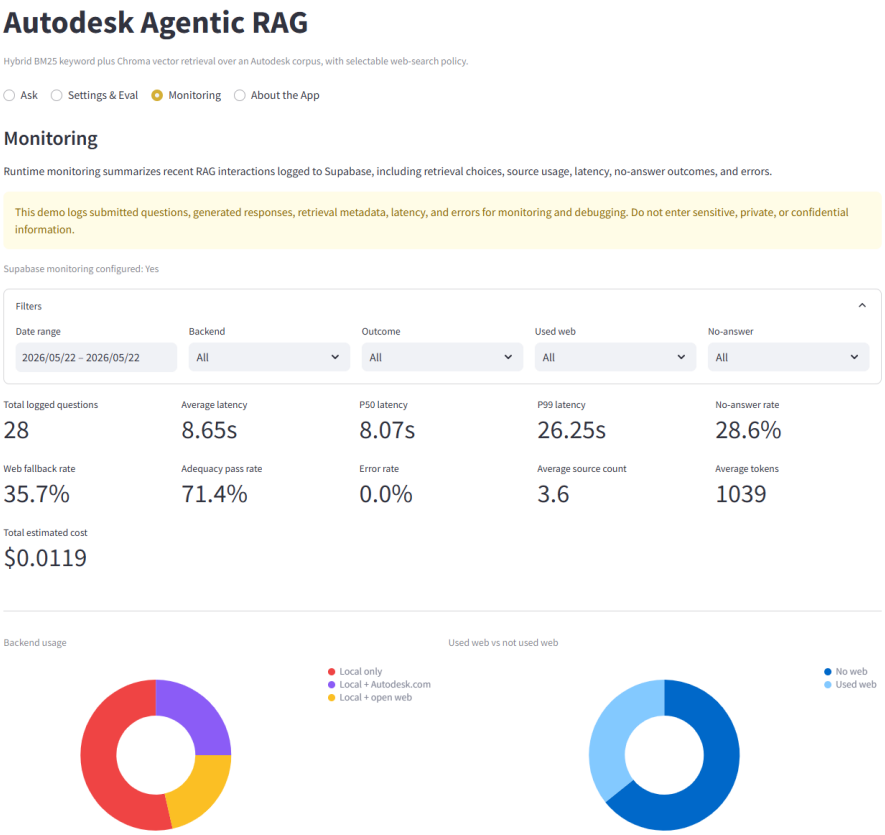
The app exposes three search policies so reviewers can compare a controlled local-corpus baseline against modes that add official Autodesk.com evidence or broader open-web corroboration.

Runtime Search Modes

Option	Search Mode	Behavior	Best Use
1	Local Document Search	Uses only local Autodesk corpus chunks from Chroma and BM25.	Fast controlled baseline.
2	Local Document Search + Autodesk.com	Combines local retrieval with official Autodesk.com web evidence.	Preferred demo mode for current official Autodesk facts.
3	Local Document Search + Open Web Search	Combines local retrieval with capped open-web evidence.	Broader corroboration when official-only search may miss context.

Evaluation & Monitoring

The project uses a fixed 50-question golden dataset and LangSmith experiments to score faithfulness, answer relevance, context precision, context recall, and latency for each search mode. The Streamlit evaluation tab can create or reuse the LangSmith dataset, run selected modes, and cache the resulting metrics for display.



Monitoring tab showing logged interaction diagnostics and retrieval metadata

Evaluation Results

All three search modes were evaluated against the same 50-question test set. Option 2 is the strongest overall portfolio demo mode because it improves answer relevance and context recall with official Autodesk.com evidence while keeping source authority more controlled than open-web search.

LangSmith Evaluation Summary

Option	Search Mode	Faithfulness	Answer Relevance	Context Precision	Context Recall	Avg Latency	P50	P99
1	Local Document Search	0.91	0.66	0.72	0.58	6.31s	6.55s	15.11s
2	Local + Autodesk.com	0.87	0.85	0.81	0.74	12.99s	12.43s	22.87s
3	Local + Open Web Search	0.84	0.78	0.86	0.62	14.11s	12.34s	73.17s

Key Insights

- Official web evidence improved answer quality: Autodesk.com search raised answer relevance and context recall over local-only retrieval.
- Local-only retrieval remained the fastest and most faithful: Option 1 had the lowest latency and highest faithfulness score.
- Open-web search added context but increased uncertainty: Option 3 had the best context precision, but weaker authority control and a much higher P99 latency.
- Corpus cleaning was essential: aggressive HTML cleanup reduced raw text volume by approximately 98.87% while preserving product-specific Markdown evidence.
- RAG safety needs multiple layers: the app combines deterministic security screening, sanitized retrieval queries, untrusted-context handling, and an adequacy gate.

Applied AI Engineering Value

This project demonstrates how to build a public, reviewable RAG application for a realistic product-support domain where answers need to be concise, grounded, current when needed, and inspectable by technical reviewers.

- Production-minded retrieval: combines local vectors, lexical search, rank fusion, source caps, context expansion, and reranking.
- Operational deployment: packages the app with Docker for Hugging Face Spaces and keeps raw local artifacts out of the hosted build context.
- Evaluation discipline: uses a fixed golden dataset and LangSmith metrics instead of relying on ad hoc demo questions.
- Monitoring path: supports Supabase-backed logging of latency, source counts, web fallback behavior, answerability outcome, and model metadata.

Project Presentation

The presentation below summarizes the project motivation, architecture, retrieval strategy, evaluation, and deployment path.

https://1drv.ms/p/c/5215a89a0899002e/IQRCHOvy_ViASaHrU7Ux9dnvASPRKAL5pzu2AFVcakNdQh4

GitHub Repository & Live Demo

The full implementation and hosted Hugging Face Space are available through the links below.

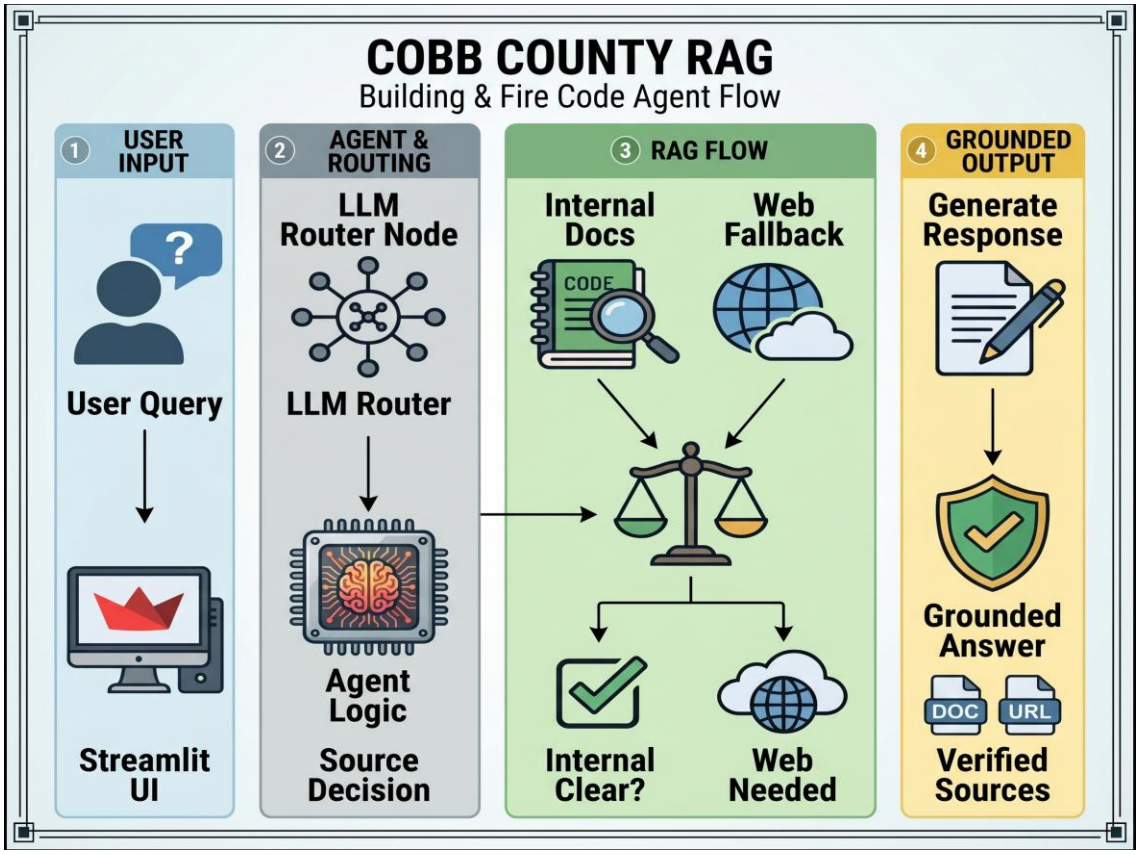
View Project Repository on GitHub

Open Live Demo on Hugging Face Spaces

The repository includes the Streamlit app, core agent workflow, retrieval modules, corpus cleaning scripts, index-building notebooks, LangSmith evaluation workflow, monitoring hooks, Docker deployment files, and documentation for reproducing the pipeline.

Disclaimer: this is a portfolio demonstration and educational project. Autodesk product details, pricing, licensing, and availability may change, so users should verify official requirements directly with Autodesk.

Cobb County Building & Fire Code Agentic RAG Assistant



Cobb County building and fire code agentic RAG project architecture flowchart (summary)

Agentic RAG • LangChain • Chroma • BM25 Hybrid Search • Query Expansion • PyPDF • Docling • Streamlit • Docker • LangSmith

Quick Facts

Role	Machine Learning Engineer / RAG Developer
Domain	Local government, building permits, fire code, public-sector document search
Tech Stack	Python, Streamlit, LangSmith/LangChain, Chroma, local BM25 corpus, PyPDF, Docling, OpenAI, SerpAPI, Docker, Databricks
Corpus	41 local PDFs, approximately 60 MB, 4,093+ loaded pages, and 13,844 vector chunks with source, parser, chunk, and page metadata
Retrieval Options	PyPDF + Chroma, Docling + Chroma, Docling + Chroma + BM25 hybrid, and Docling + query expansion + BM25 hybrid
Recommended Default	Option 3: Docling + Chroma + BM25 Hybrid Search, based on the strongest balanced retrieval metrics
Live Demo	https://cobb-county-code-rag-assistant.streamlit.app/

This project is an agentic retrieval-augmented generation (RAG) application for Cobb County, Georgia building and fire code questions. It uses local PDF documents as the primary knowledge base, supports four retrieval configurations, and falls back to official web sources when local evidence is weak or when a question asks about current, adopted, recently changed, or effective-date information.

The system demonstrates production-minded RAG engineering: document ingestion, selectable retrieval backends, vector and hybrid search, LLM routing, deterministic context expansion, strict evidence gating, source-grounded synthesis, web verification, LangSmith evaluation, Docker support, and a recruiter-friendly Streamlit interface.

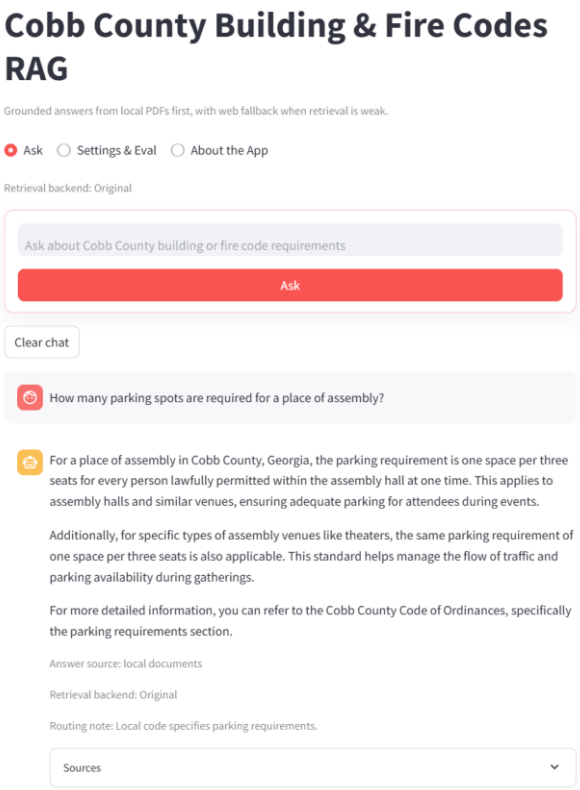
Background & Problem Statement

Building and fire code information is difficult to search because it is spread across ordinances, code PDFs, county forms, permit guidance, plan-review checklists, Fire Marshal documents, and state-level code references. A homeowner, contractor, reviewer, or administrative user may know the question they want to ask, but not the exact document, section, or page where the answer appears.

Problem Statement: Can a deployable RAG assistant help users ask natural-language questions about Cobb County code and permitting documents while keeping answers grounded in cited local documents and current web evidence when verification is needed?

Streamlit Chat Interface

The front end is a Streamlit chat application designed for portfolio review and technical inspection. Users submit a code or permitting question, then receive a concise answer with source references and an answer-source label. The Settings & Eval dashboard lets users switch between all four retrieval options and view cached LangSmith evaluation metrics without restarting the app.



Streamlit user interface for the Cobb County building and fire code RAG assistant

Document Corpus & Data Policy

The local corpus is designed around Cobb County, Georgia building and fire code research. It includes county ordinance PDFs, Fire Marshal forms and checklists, building permit and tenant build-out guidance, fire inspection documents, hydrant and sprinkler resources, emergency equipment guidance, and Georgia code references.

- Local PDFs used during development: 41 files.
- Raw PDF size: approximately 60 MB.
- Loaded pages: 4,093+ pages.
- Vector chunks: 13,844 chunks stored in Chroma.
- Repository policy: raw PDFs are excluded, while the generated vectorstore and BM25 artifacts are tracked with Git LFS for deployment.

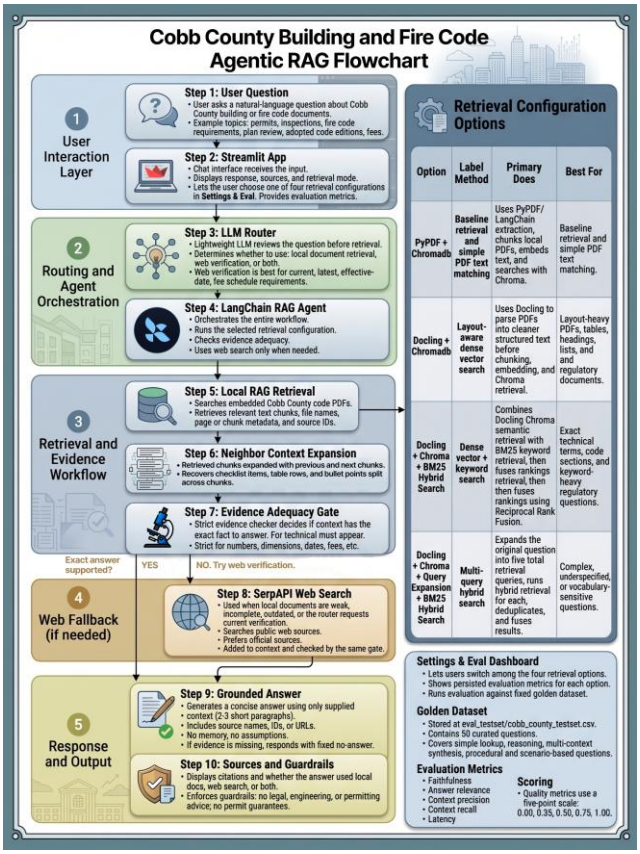
Supported Retrieval Backends

Backend	UI Label	Retrieval Strategy
cobb_code_docs_original	Option 1: PyPDF + Chromadb	Original PyPDF/LangChain page extraction with Chroma vector search
cobb_code_docs_docling	Option 2: Docling + Chromadb	Docling layout-aware Markdown conversion with Chroma vector search
docling_chroma_bm25_hybrid	Option 3: Docling + Chroma + BM25 Hybrid Search	Docling chunks searched with both Chroma vector retrieval and a persisted local BM25 keyword corpus
docling_chroma_bm25_expansion	Option 4: Docling + Query Expansion + BM25 Hybrid Search	Reuses Option 3 indexes, expands the original question into five retrieval queries, deduplicates results, and fuses rankings

Agentic RAG Architecture

The app uses an agentic RAG workflow rather than a single prompt-only LLM call. A lightweight LLM router evaluates the question, the selected retrieval backend searches local evidence, deterministic neighbor expansion keeps nearby checklist items visible, an evidence check determines whether the local context is sufficient, and web search is triggered when current-code verification or weak local evidence requires it.

- Settings selector: switches new questions across four retrieval configurations without requiring an app restart.
- Query routing: flags whether the question needs local documents, web search, or both.
- Local retrieval: uses Chroma vector search or Chroma + BM25 hybrid retrieval depending on the selected option.
- Query expansion: Option 4 uses one LLM call to create four additional retrieval queries before hybrid retrieval.
- Deterministic context expansion: expands each retrieved chunk with same-document neighboring chunks chunk_index - 1 and chunk_index + 1 .
- Strict evidence gate: requires exact supporting evidence for numeric, code, inspection, permit, and procedural answers.
- Web fallback: prioritizes public county and state web sources for current-code verification.



Agentic RAG flowchart showing local retrieval, evidence check, web fallback, and grounded response generation

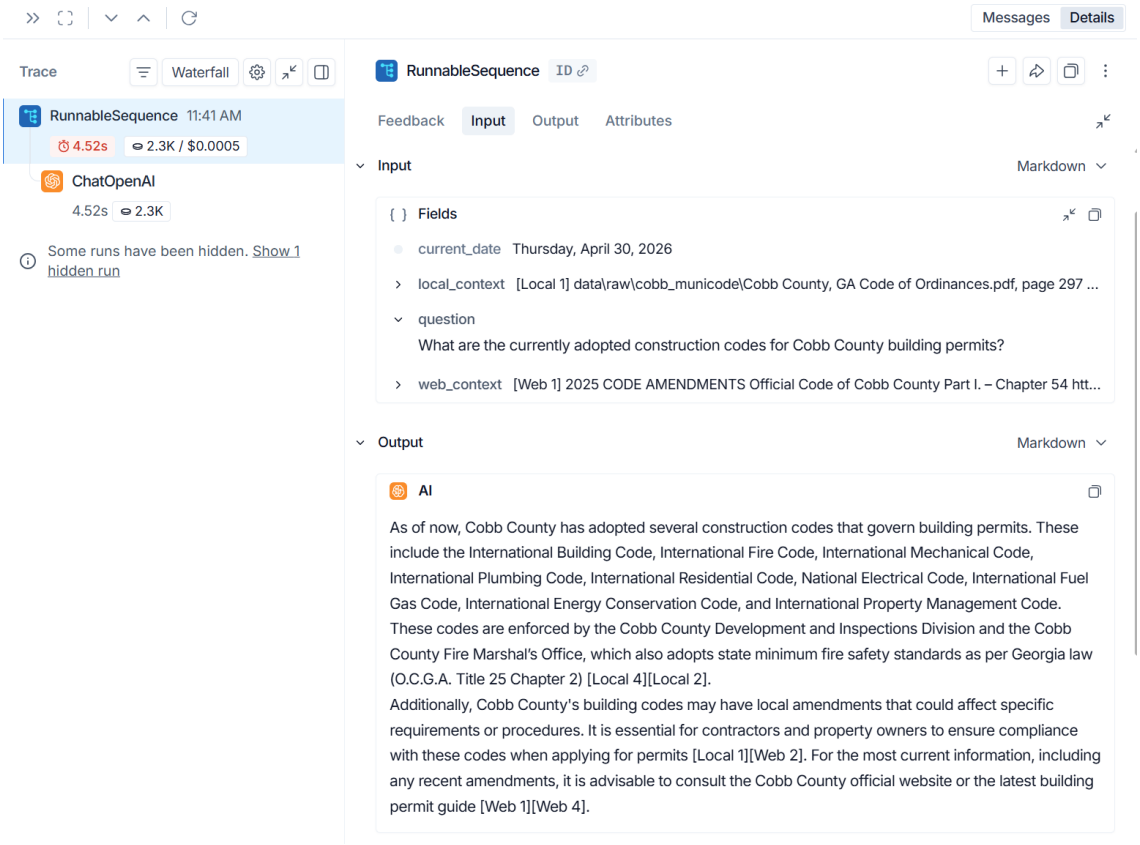
Retrieval Engineering

This project does not train a conventional tabular ML model. Instead, the engineering work focuses on building reliable retrieval over long, heterogeneous PDF documents where exact numbers, section references, conditions, and procedural language matter. The final app compares vector-only retrieval against local hybrid retrieval and query-expansion retrieval.

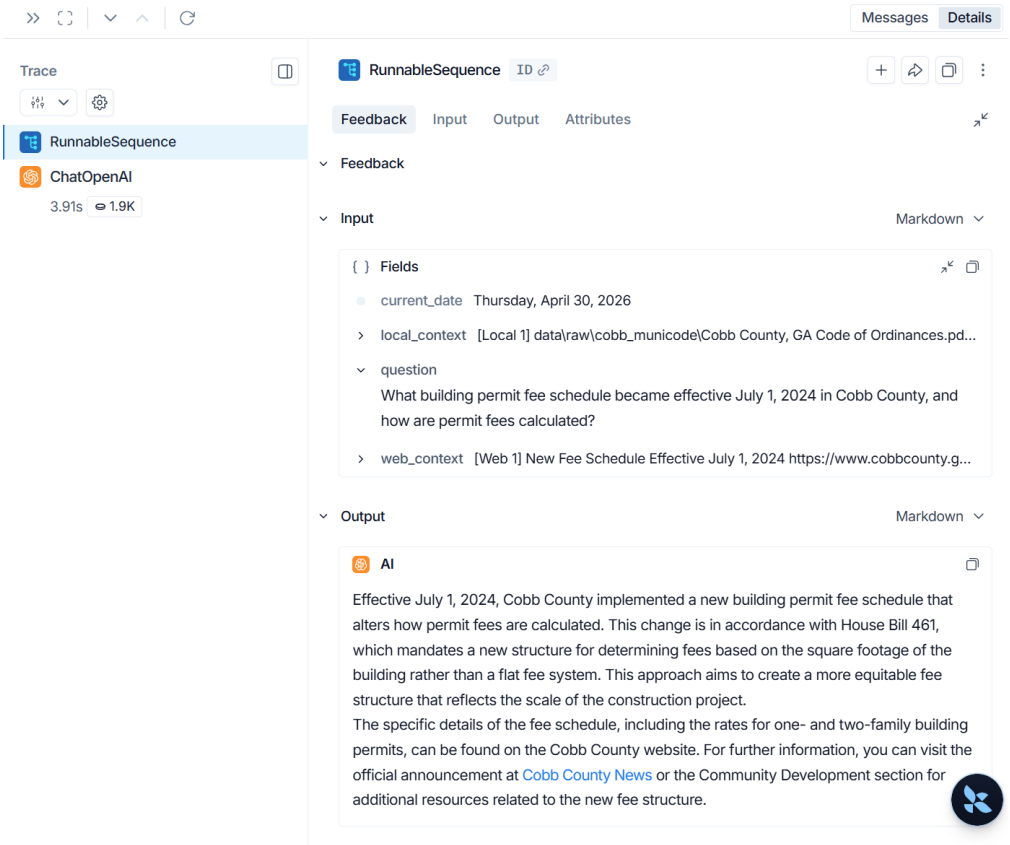
- Original parsing: preserves the initial PyPDF/LangChain page extraction pipeline.
- Docling parsing: converts layout-aware PDF content to Markdown before chunking, which can help with headings, tables, sections, and multi-column documents.
- BM25 hybrid retrieval: combines keyword matching with dense vector retrieval using Reciprocal Rank Fusion.
- Query expansion: generates four additional technical and step-back queries, retrieves with all five queries, deduplicates, and fuses rankings.
- Context expansion: retrieves small chunks for search quality, then expands each hit with only same-document neighboring chunks before adequacy checking.
- Metadata tracking: file name, source path, parser type, backend, chunk index, section, and page range are retained where available.
- Evidence thresholding: weak retrieval or current-code language can trigger web search or conservative abstention.

LangSmith Tracing, Evaluation & Validation

LangSmith traces were used to inspect the full chain input, routing context, local PDF evidence, web-search evidence, token usage, latency, and final answers. The project uses a fixed 50-question golden evaluation set and cached LangSmith experiment scores for faithfulness, answer relevance, context precision, context recall, and latency.



LangSmith trace showing local document and web context used by the RAG assistant



LangSmith trace showing web search fallback for a Cobb County fee schedule question

Evaluation Methodology

The Settings & Eval dashboard compares all four retrieval configurations using the same fixed 50-question golden dataset. Metrics are persisted under eval_results/ so the dashboard can load results immediately without automatically launching a new evaluation run.

- Faithfulness: whether all answer claims are supported solely by the retrieved context.
- Answer relevance: whether the generated response directly answers the user's question.
- Context precision: whether retrieved context is useful and well-ranked rather than noisy.
- Context recall: whether the retrieved context contains the necessary facts compared with the ground-truth answer.
- Latency: mean, P50, and P99 runtime in seconds across the fixed evaluation set.

Results & System Checks

The project was validated through ingestion checks, retrieval smoke tests, current-code routing tests, web search fallback tests, syntax checks, Docker support, manual review of representative questions, and LangSmith evaluation across four retrieval options.

Retrieval Method Performance Comparison

Option	Method	Answer Relevance	Context Precision	Context Recall	Faithfulness	Avg Latency (s)	P50 (s)	P99 (s)
1	PyPDF + Chromadb	0.695	0.695	0.610	0.940	15.3	10.4	108.8
2	Docling + Chromadb	0.625	0.650	0.615	0.930	13.6	12.2	36.1
3	Docling + Chroma + BM25 Hybrid Search	0.725	0.765	0.720	0.955	15.4	11.4	69.0
4	Docling + Query Expansion + BM25 Hybrid Search	0.730	0.755	0.645	0.965	23.4	21.3	50.2

Best balanced configuration: Option 3 is the strongest practical default because it produces the best context precision and context recall while avoiding the extra query-expansion latency of Option 4. Highest faithfulness: Option 4 reaches the top faithfulness score, but the added query-expansion LLM call increases median latency and does not improve context recall over Option 3.

Validation Summary

Test Area	Result	Notes
PDF ingestion	Passed	Loaded Cobb County and Georgia code PDFs
Vector index build	Passed	Indexed 13,844 chunks into Chroma
Docling-enhanced indexing	Added	Builds cobb_code_docs_docling from Docling Markdown output
BM25 hybrid retrieval	Added	Persists a local BM25 corpus and fuses keyword and vector rankings
Query expansion retrieval	Added	Creates five total retrieval queries before hybrid ranking
Deterministic context expansion	Added	Expands retrieved chunks with same-document -1/+1 neighbors before evidence checks
Strict evidence gate	Added	Requires exact support before numeric, code, inspection, permit, or procedural answers

Test Area	Result	Notes
LangSmith evaluation cache	Added	Saves per-backend metrics under eval_results/
Web fallback	Passed	SerpAPI Google Search works from the app environment
Deployment support	Included	Streamlit Community Cloud, Dockerfile, and docker-compose.yml

Key Insights

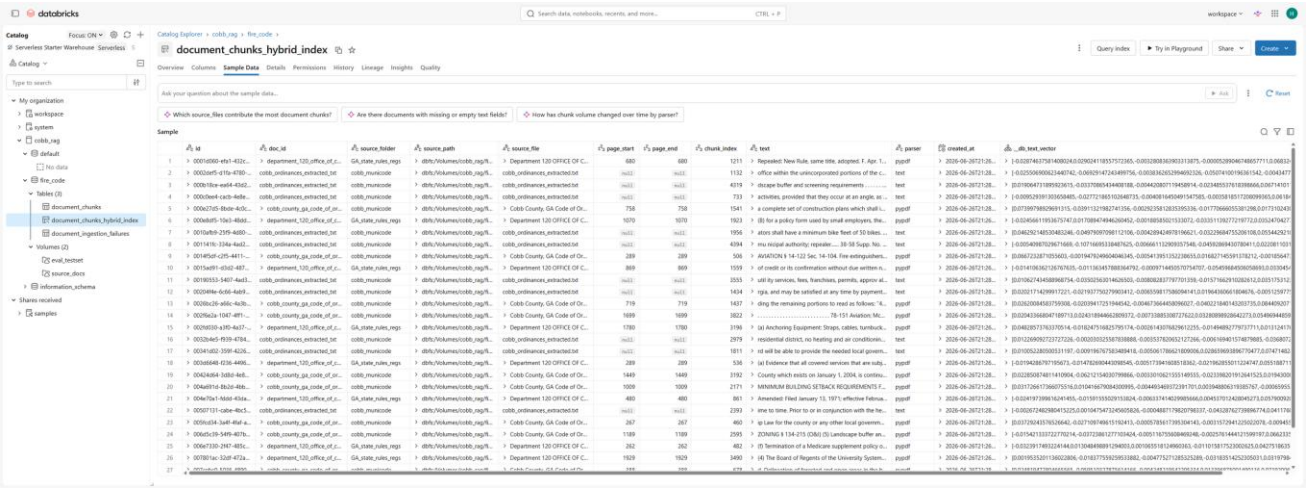
- Hybrid retrieval mattered most: Docling alone did not outperform the PyPDF baseline, but Docling paired with BM25 improved context precision and recall.
- Faithfulness improved through stricter evidence gating: the app refuses numeric, code, permit, inspection, and procedural answers unless exact support is visible in the supplied context.
- Neighbor expansion reduced false refusals: adding same-document neighboring chunks helps when a retrieved chunk lands just before an answer-bearing checklist item or table row.
- Query expansion has a cost: Option 4 produced the highest faithfulness and answer relevance, but at a substantially higher median latency.
- Recommended demo configuration: Option 3 gives the best retrieval-quality tradeoff for portfolio demonstrations.

Applied ML Engineering Value

This project demonstrates how modern LLM systems can be adapted to document-heavy public-sector workflows where answers must be grounded, current, and easy to verify. It is especially relevant to customer-facing AI platforms, conversational question answering, search and retrieval, agent workflows, and intelligent automation.

- RAG system design: combines selectable retrieval backends, local retrieval, hybrid search, routing, evidence checking, and web fallback.
- Operational grounding: returns source references instead of unsupported open-ended answers.
- Production mindset: includes reproducible ingestion, Docker support, environment templates, Git LFS artifacts, and hosted app deployment.
- Evaluation and observability: uses LangSmith traces and fixed golden-set metrics to inspect retrieval quality, routing decisions, latency, and generated outputs.

Databricks Implementation



Databricks AI Search hybrid index for Cobb County building and fire code document chunks

I also implemented a Databricks version of the app, adapting the local RAG workflow to a managed lakehouse deployment. Source documents are loaded from a Unity Catalog volume, chunked into a managed Delta table, indexed through Databricks AI Search hybrid retrieval, and served through a Databricks Streamlit app.

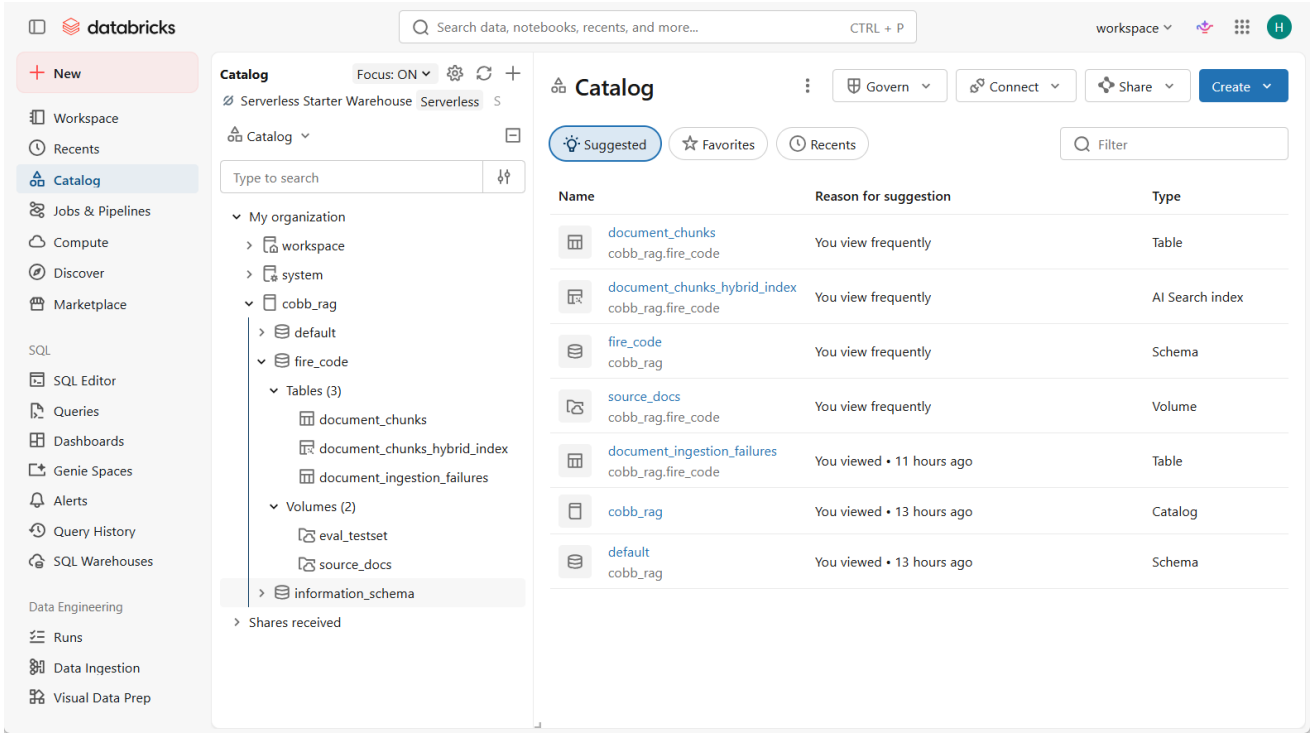
The Databricks pipeline preserves the important production behaviors from the original app where practical: hybrid document retrieval, same-document neighbor context expansion, CrossEncoder reranking, source-grounded answer generation, and evaluation against the same fixed 50-question golden test set.

- Lakehouse storage: documents and generated chunks are organized in Unity Catalog assets for governed retrieval workflows.
- Managed search: the local Chroma/BM25 layer is replaced with a Databricks AI Search hybrid index over the chunk table.
- Deployment target: the Streamlit interface is packaged as a Databricks app for workspace-hosted querying.
- Evaluation parity: Databricks RAG evaluation is performed using the same 50-question benchmark.

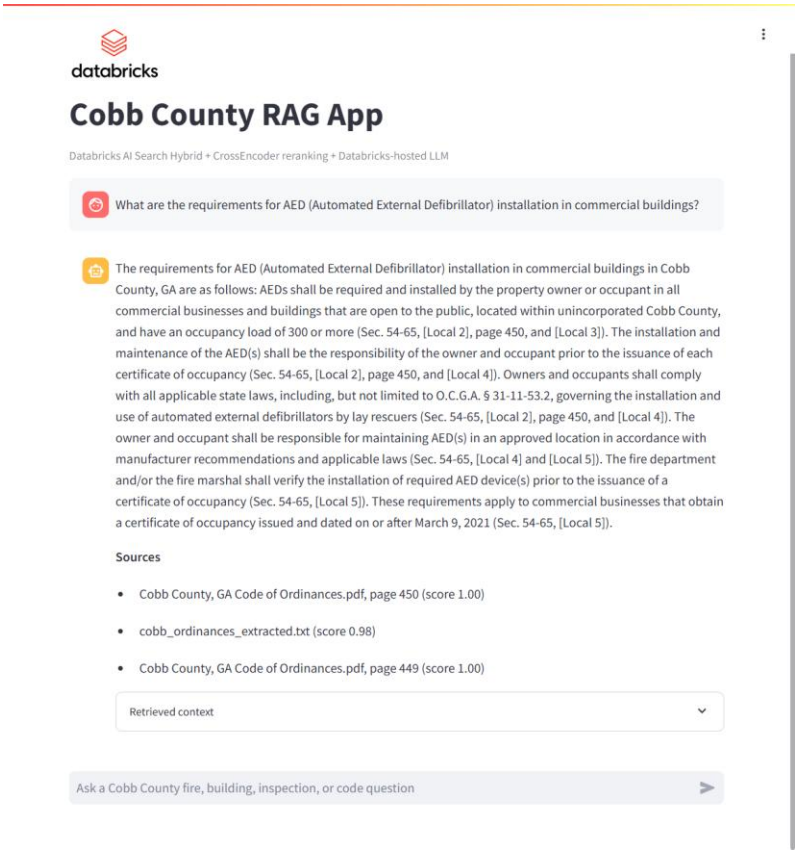
Databricks Evaluation Performance

Setting	Questions	Faithfulness	Answer Relevance	Context Precision	Context Recall	Avg Latency (s)	P50 (s)	P99 (s)
Databricks AI Search Hybrid + CrossEncoder	50	0.920	0.880	0.875	0.855	9.69	8.92	24.67

The Databricks deployment maintained strong grounding and retrieval quality while moving storage, indexing, and serving into managed platform services. Average latency was 9.69 seconds, with 8.92 seconds at P50 and 24.67 seconds at P99, reflecting the managed app and model-serving path plus occasional heavier retrieval or generation cases.



Unity Catalog assets for the Databricks Cobb County RAG implementation



Databricks Streamlit app example query for the Cobb County building and fire code RAG assistant

GitHub Repository & Live Demo

The full implementation and hosted Streamlit demo are available through the links below.

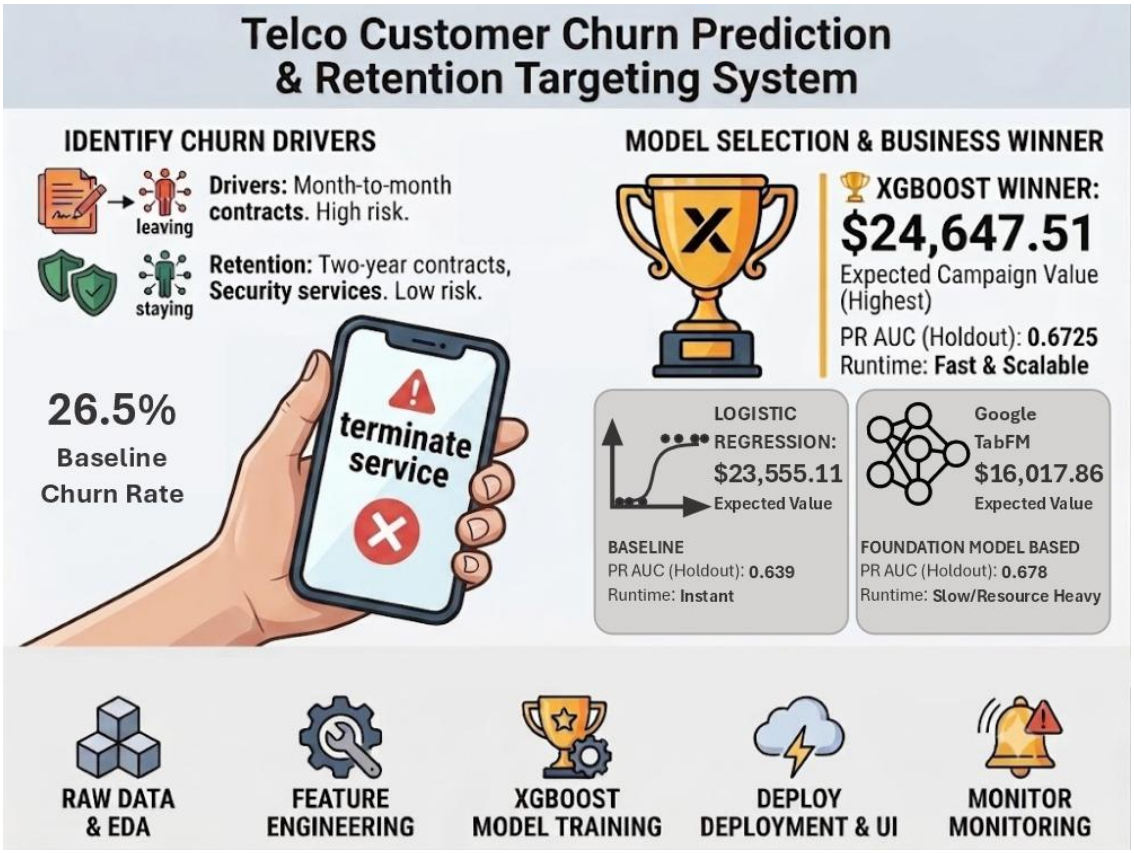
🔗 View Project Repository on GitHub

🚀 Open Live Demo

The repository includes the Streamlit app, LangChain agent workflow, Chroma retriever, BM25 hybrid retrieval, query expansion retrieval, PyPDF and Docling ingestion scripts, deterministic context expansion, LangSmith evaluation workflow, Docker files, environment template, and documentation for rebuilding local vector and BM25 indexes.

Disclaimer: this is a portfolio demonstration and educational project. It is not legal, engineering, building code, fire code, or permitting advice. Users should verify requirements directly with Cobb County, the Georgia Department of Community Affairs, the State Fire Marshal, and the authority having jurisdiction.

Telco Customer Churn Prediction & Retention Targeting System



Telco customer churn prediction and retention targeting workflow overview

Customer Churn Prediction • Retention Targeting • XGBoost • Logistic Regression • TabFM Benchmark • Optuna • MLflow • FastAPI • Streamlit • Great Expectations • Docker • Hugging Face Spaces

Quick Facts

Role	Data Scientist / Machine Learning Engineer
Domain	Telecommunications, subscription analytics, customer retention strategy
Objective	Predict churn risk and rank customers by expected retention-campaign value
Dataset	IBM Telco Customer Churn dataset with 7,043 records and 33 customer, service, billing, and churn-related fields
Recommended Model	Optuna-tuned XGBoost selected for the practical retention-targeting workflow
Repository	github.com/helsharif/Customer-Churn-End-to-End-ML
Live Demo	huggingface.co/spaces/helsharif/telco-churn-prediction

This project builds a production-oriented machine learning workflow for predicting customer churn in a fictional telecommunications business. The system moves beyond a notebook-only classifier by connecting leakage-safe feature engineering, model selection, experiment tracking, API serving, a Streamlit user interface, local monitoring, Docker packaging, and a Hugging Face Spaces deployment.

The business goal is not simply to maximize accuracy. The final workflow ranks customers by expected campaign value, combining churn probability with retained lifetime value and intervention cost assumptions so that retention teams can prioritize outreach where it is most likely to pay off.

Background & Problem Statement

Customer churn reduces recurring revenue and increases customer-acquisition pressure. In a subscription telecom setting, churn risk is shaped by contract terms, tenure, service bundles, billing behavior, support options, and price sensitivity. A useful model needs to identify customers likely to leave early enough for the business to intervene.

Problem Statement: Given customer demographics, subscribed services, account tenure, contract details, and billing history, can a reproducible ML system predict quarterly churn and translate those probabilities into a ranked retention-campaign list?

Dataset & Leakage Controls

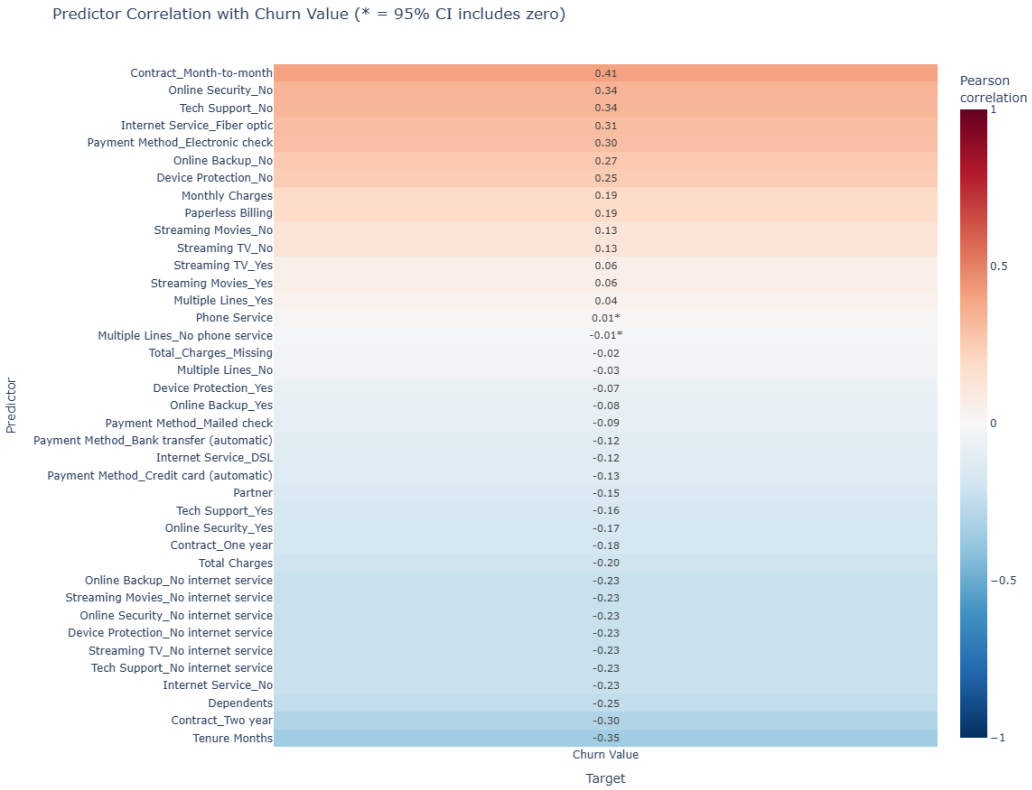
The raw source is the IBM Telco Customer Churn dataset, which contains 7,043 customer records and 33 variables covering demographics, geography, service subscriptions, contract and billing information, tenure, churn outcome, and churn explanation fields.

The preferred modeling target is Churn Value , where 1 indicates churn and 0 indicates retention. The pipeline excludes direct identifiers, direct target equivalents, raw coordinates, and post-outcome churn explanation fields from model features.

- Leakage exclusions: CustomerID, Count, Lat Long, Latitude, Longitude, Churn Label, Churn Value, Churn Score, Churn Reason, Churn Category, and similar churn-derived fields.
- Feature handling: yes/no fields are encoded as 0/1, compact categorical variables are one-hot encoded, and blank Total Charges values are handled safely.
- Model-ready outputs: the workflow creates separate datasets for logistic regression and XGBoost/TabFM comparison so each model family can use an appropriate feature set.

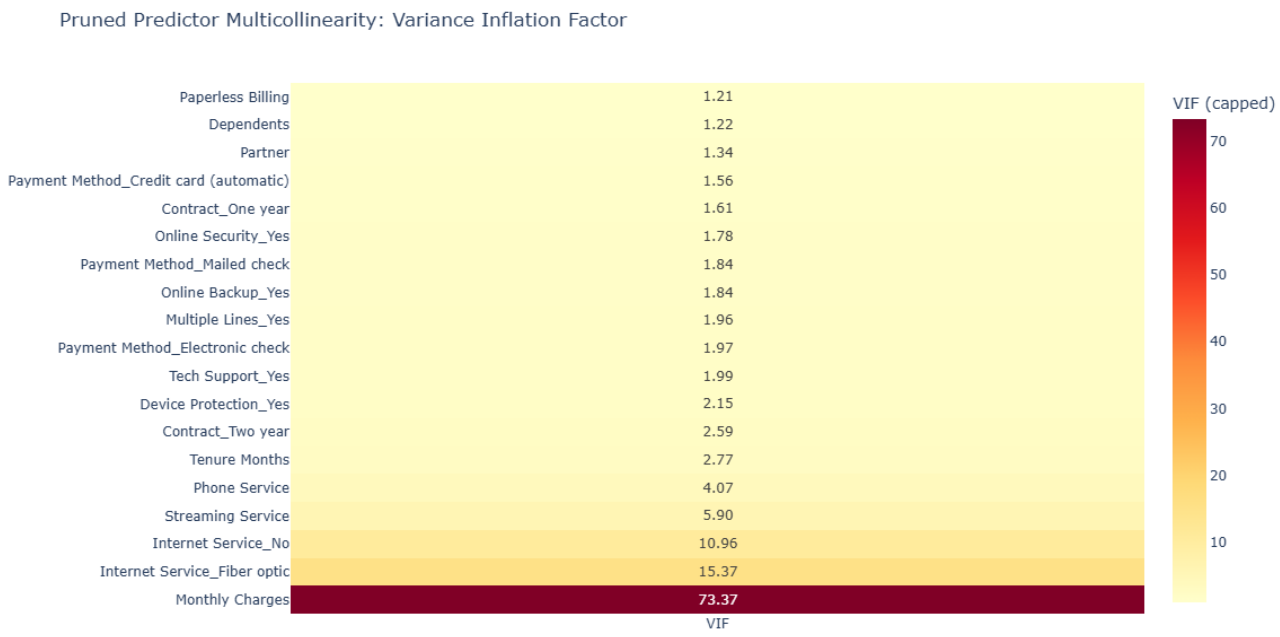
Exploratory Analysis & Feature Review

Exploratory analysis focused on understanding which encoded predictors were associated with churn and which predictors carried redundant signal. These checks informed the interpretable baseline while preserving a richer feature set for the nonlinear XGBoost workflow.



Predictor correlations with churn in the encoded telco modeling dataset

Positive correlations indicate predictors associated with higher churn risk, while negative correlations indicate predictors associated with retention. The multicollinearity review helped prune redundant inputs for the logistic-regression baseline; XGBoost retained a broader encoded feature set because tree-based models are less sensitive to linear multicollinearity.



Multicollinearity review for churn predictors using VIF

Modeling Approach

The research notebooks compare logistic regression, Optuna-tuned XGBoost, and a PyTorch/CUDA TabFM benchmark using consistent stratified holdout logic. The packaged production-style workflow focuses on the two practical model families: logistic regression and XGBoost.

- Logistic Regression: a transparent baseline with directly inspectable coefficients and very fast training.
- XGBoost: a tuned gradient-boosted tree model that captures nonlinear interactions and supports SHAP-style interpretation.
- TabFM benchmark: a transformer-based tabular foundation-model experiment retained as historical reference, but not included in the serving and monitoring workflow.
- Thresholding: candidate notebooks default to the training-set churn rate, with validation-based threshold tuning in the reusable package pipeline.
- Experiment evidence: MLflow logs lightweight parameters, metrics, comparison tables, prediction outputs, and retention-targeting summaries.

Model Trade-Offs

Model	Role	Strength	Best Use
Logistic Regression	Transparent baseline	Fast, auditable, coefficient-based interpretation	Fallback when governance or simplicity matters most
XGBoost	Recommended practical model	Strong nonlinear ranking and best top-100 campaign value	Production-style retention targeting workflow
TabFM	Foundation-model benchmark	Strong conventional metrics in historical testing	Experimental comparison, not the deployed workflow

Model Selection & Retention Value

Historical model-selection experiments used five-fold stratified cross-validation on the training split, then evaluated candidates once on the untouched holdout set. TabFM achieved the strongest conventional predictive metrics in that saved run, but XGBoost was selected as the practical recommendation because it delivered the highest expected net value for the top-100 retention campaign list.

Saved Model-Selection Evidence

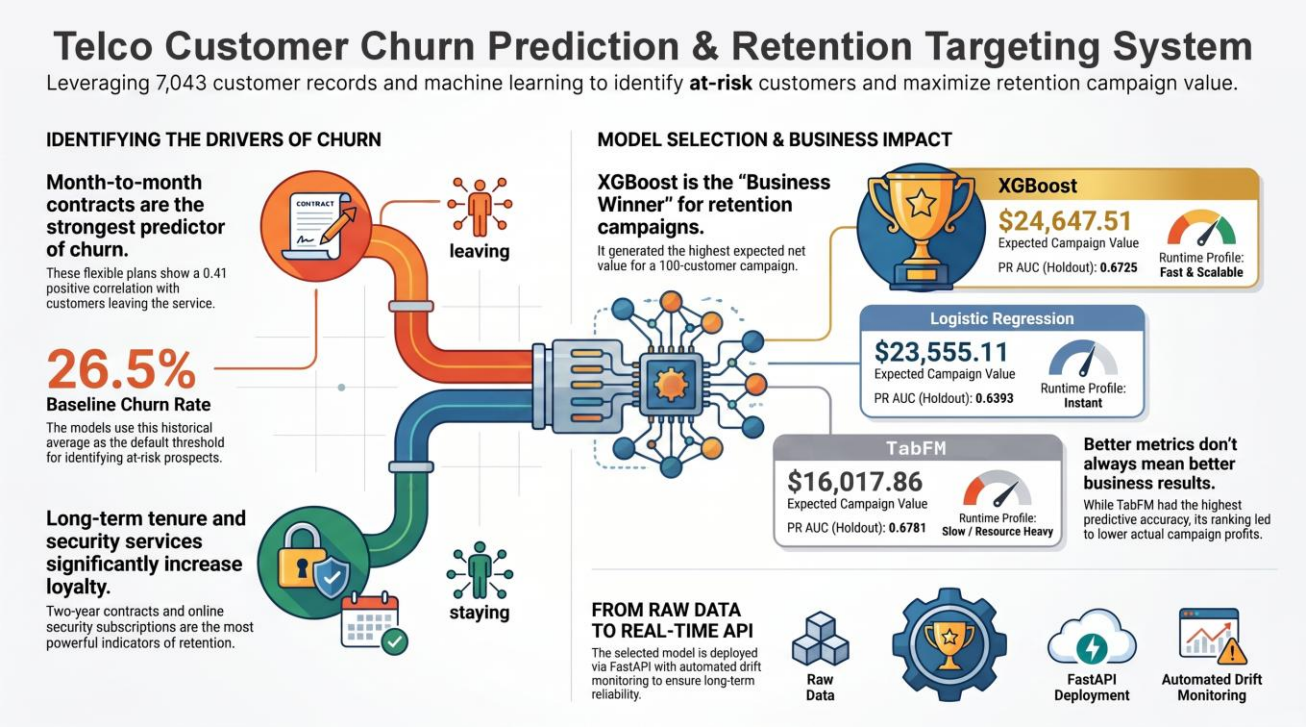
Model	Holdout PR AUC	Holdout ROC AUC	Top-100 Expected Net Value
Logistic Regression	0.6393	Baseline reference	$23,555.11
XGBoost	0.6725	0.8559	$24,647.51
TabFM	0.6781	0.8599	$16,017.86

This result illustrates the difference between general ranking quality and campaign value. Retention targeting combines churn probability, retained-LTV assumptions, offer acceptance, retention uplift, outreach cost, and offer cost. XGBoost was nearly tied with TabFM on PR AUC, ran much faster, and produced the stronger value-ranked outreach list under the saved business assumptions.

Production-Oriented MLOps Workflow

The project packages the reusable logic under src/churn_ml/ so training, preprocessing, prediction, serving, monitoring, and UI interaction are not trapped inside notebooks. The production-style pipeline compares logistic regression and XGBoost, selects by validation PR AUC, tunes the threshold, evaluates once on the test split, and persists a local production model bundle.

- Validation and preprocessing: raw schema checks, leakage-safe feature generation, and documented processed outputs.
- Training pipeline: reproducible model comparison, MLflow tracking when available, saved metrics, and persisted model artifacts.
- API serving: FastAPI endpoints for health checks, model information, and churn predictions.
- Monitoring foundation: anonymized JSONL inference logs and local drift checks against a saved training baseline.
- CI/CD readiness: pytest, Ruff linting, import validation, and Docker image build checks through GitHub Actions.

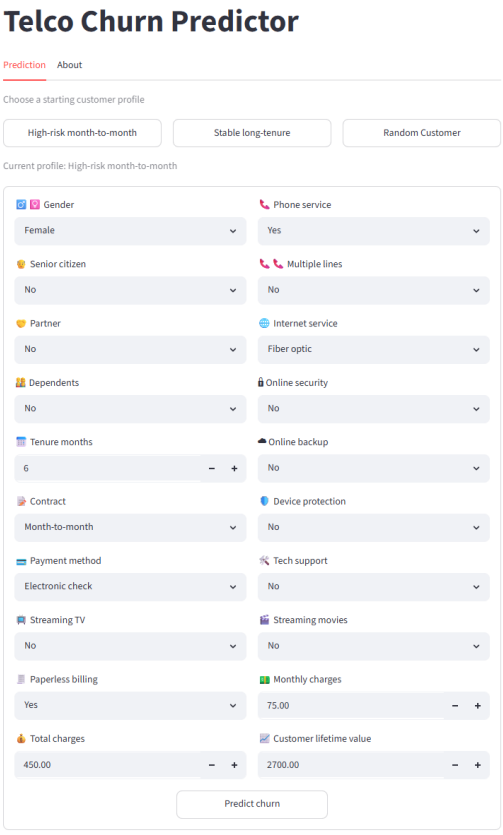


End-to-end telco churn ML workflow from raw data through deployment and monitoring

Streamlit Prediction App

The live application uses a Streamlit front end against a FastAPI prediction endpoint. In the Hugging Face Spaces deployment, the Docker image trains or loads the production model artifact, runs FastAPI internally, and exposes the Streamlit UI on the public Space port.

- Interactive inputs: customer profile, service, contract, billing, and tenure fields are converted into model-ready features.
- Prediction output: the app returns churn probability, classification, and retention-targeting context for the submitted profile.
- Deployment architecture: Docker keeps the hosted demo aligned with the local FastAPI plus Streamlit workflow.



Streamlit interface for the telco churn prediction and retention targeting app

Business & Portfolio Value

The project demonstrates how to connect model quality to a business-facing retention decision. Rather than stopping at classification metrics, it turns churn predictions into a campaign list that can be evaluated against cost, uplift, and customer-value assumptions.

- Actionable targeting: prioritizes customers by expected campaign value, not just raw churn risk.
- Governance awareness: excludes leakage fields and keeps interpretable logistic-regression evidence available alongside the stronger XGBoost candidate.
- Deployment realism: includes API serving, UI interaction, Docker packaging, monitoring foundations, and CI checks.
- Clear trade-off analysis: explains why the production-style workflow uses XGBoost even though TabFM was a useful benchmark.

GitHub Repository & Live Demo

The full implementation is available on GitHub, and the public Hugging Face Space hosts the Streamlit churn prediction interface backed by the production-style FastAPI workflow.

View Project Repository on GitHub

Open Live Demo on Hugging Face Spaces

The implementation includes leakage-safe preprocessing, logistic-regression and XGBoost model comparison, MLflow experiment evidence, FastAPI serving, Streamlit UI interaction, Docker deployment support, drift-check scaffolding, and CI/CD-ready project structure.

AI-Automated Lead Generation Pipeline for Engineering Consultancy (RAG + Web Data)

AI-automated lead generation dashboard for engineering consultancy

RAG • Web Scraping • Lead Scoring • OpenAI GPT-4o • Gemini 2.0 Flash • Vector Databases • Sales Enablement

Quick Facts

Role	Lead Data Scientist / AI Consultant
Domain	B2B Lead Generation, Environmental Engineering, Sales Operations
Tech Stack	Python, Pandas, Web Scraping, OpenAI GPT-4o, Gemini 2.0 Flash, Vector DB (RAG), REST APIs
Stakeholder	Mid-Sized Environmental Engineering Consultancy

This project built an AI-enabled lead-generation pipeline to identify, rank, and summarize high-value prospects for a mid-sized environmental engineering consultancy. Historically, the business relied on manual Google searches, conferences, and word-of-mouth referrals, which were time-consuming and often missed promising opportunities.

By combining web data, regulatory databases, and retrieval-augmented generation (RAG) , the pipeline systematically discovers "ideal fit" prospects and produces context-rich briefs that sales teams can plug directly into CRMs and targeted outreach campaigns.

Background & Problem Statement

Environmental and engineering consultancies often serve narrow industry niches, where the "right" clients are those with complex operations, regulatory exposure, and recurring compliance needs. Yet traditional lead-generation approaches - cold lists, generic web searches, or conference networking - are labor-intensive and difficult to scale .

At the same time, rich data about facilities and companies is available from public web sources (company websites, news), and government databases such as the EPA's ECHO system, which track permits, violations, and enforcement history.

Problem Statement: How can AI, web data, and domain-specific RAG workflows be combined to automatically discover and prioritize high-potential clients for a niche engineering consultancy, while providing enough context for targeted outreach and proposal development ?

Pipeline Design & Data Engineering

The system was designed as an end-to-end pipeline that ingests web and regulatory data, structures it for retrieval, and exposes it to LLMs via a RAG layer:

- Data Ingestion & Web Scraping: Automated collection of facility and company metadata from public sources including Google Search, the EPA's ECHO database, and company websites (e.g., sector, NAICS codes, location, permits, enforcement history).
- Normalization & Feature Engineering: Cleaned and standardized facility attributes (industry codes, permitted pollutants, violation counts, geographic region) into a structured tabular schema suitable for both analytics and RAG.
- RAG Layer & Document Store: Stored structured and semi-structured documents in a vector database to enable similarity search and grounding of LLM responses in real facility-level data, not generic model priors.
- Ideal Customer Profile (ICP): Encoded domain-specific criteria - industry type, facility size, regulatory exposure, complexity of operations, and historical compliance issues - into an explicit "ideal customer profile" that drives downstream scoring.
- Workflow Orchestration: Used Python and Pandas scripts to orchestrate data collection, RAG queries, LLM evaluations, and markdown report generation over batches of candidate facilities.

AI Ranking, Summarization & Reporting

With the data pipeline in place, LLMs were used to evaluate and narrate each prospect relative to the consultancy's ICP:

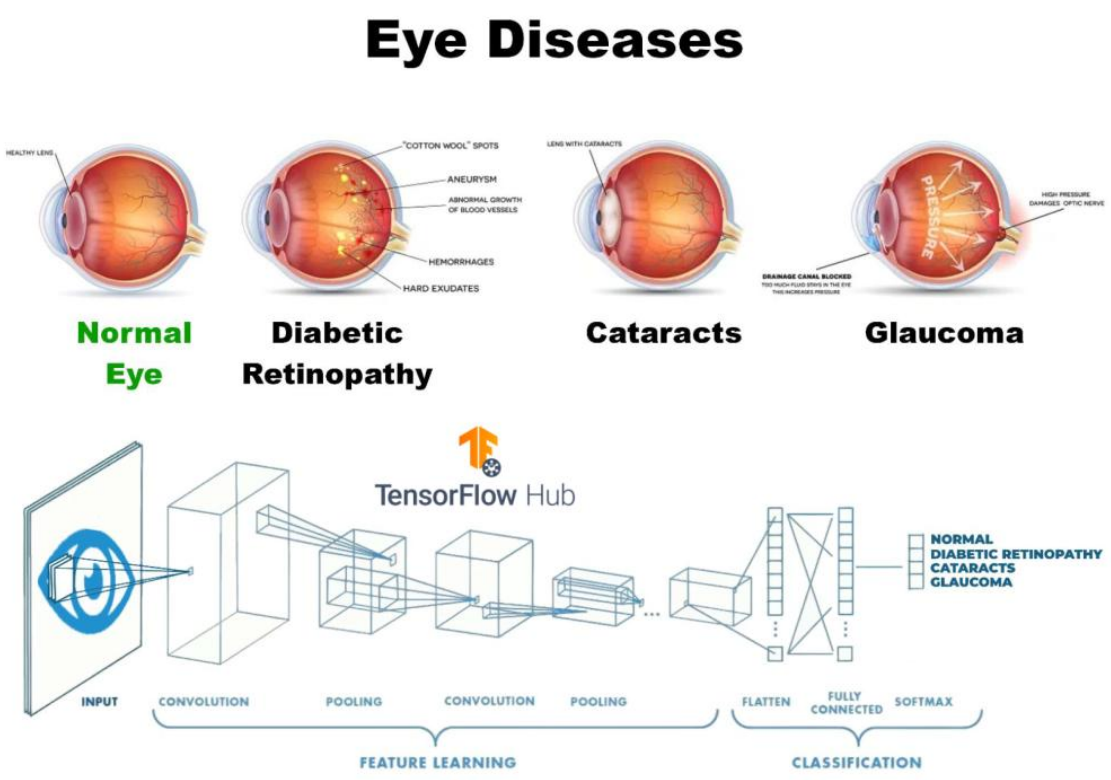
- Grounded LLM Evaluation: For each candidate facility, the RAG layer retrieves relevant documents (permits, enforcement records, company descriptions) which are fed into OpenAI GPT-4o and Gemini 2.0 Flash via structured prompts.
- Suitability Scoring: LLMs output qualitative scores such as High , Medium , or Low fit, tied to the ICP criteria (e.g., recurring monitoring needs, complex treatment trains, multi-site operations).
- Prospect Brief Generation: For high- and medium-scoring facilities, the system generates short narrative briefs summarizing why the prospect is a strong fit and which services are likely to be relevant (e.g., permitting, compliance audits, monitoring design).
- Report Packaging: Generated briefs are compiled into a markdown/HTML report that can be dropped directly into a CRM, shared with business-development teams, or exported to PDF for review.

Impact & Actionable Insights

The AI-automated pipeline substantially improved the efficiency and quality of lead generation for the consultancy:

- Order-of-Magnitude Time Savings: Reduced the effort to identify and research 50-100 high-potential prospects from several days of manual work to a few hours of automated runs.
- Discovery of "Non-Obvious" Leads: Surfaced facilities with significant regulatory exposure or complex operations that traditional word-of-mouth and conference-based approaches would likely miss.
- Sales-Ready Outputs: Delivered pre-structured prospect briefs that can be pasted directly into CRMs or used as a launchpad for personalized outreach, improving sales readiness and conversion potential.
- Reusable RAG Framework: Established a modular RAG + LLM template that can be extended to other verticals (e.g., water utilities, industrial manufacturing, healthcare labs) with minimal reconfiguration.

AI-Powered Retinal Disease Detection with Deep Learning & Computer Vision Pipeline



Schematic of Tensorflow Computer Vision Pipeline from Retinal Image to Normal/Eye Disease Classification

Schematic of Tensorflow Computer Vision Pipeline from Retinal Image to Normal/Eye Disease Classification

Computer Vision • Transfer Learning • EfficientNet • TensorFlow / Keras • Multi-Class Classification • Healthcare AI

Quick Facts

Role	Lead ML Engineer / Computer Vision Researcher
Domain	Medical Imaging, Ophthalmology, Clinical Decision Support
Tech Stack	Python, TensorFlow/Keras, EfficientNet-Bx, OpenCV, NumPy, Pandas, Matplotlib/Plotly
Dataset	4,000+ retinal fundus images (456×456 px) labeled as cataract, diabetic retinopathy, glaucoma, or normal
Live Demo	https://helsharif-retinal-disease-classifier.hf.space

This project developed a deep learning-based computer vision pipeline to automatically classify retinal fundus images into four categories: cataract , diabetic retinopathy , glaucoma , and normal . The goal is to demonstrate how AI can support early screening by flagging at-risk patients for follow-up with ophthalmologists.

Using transfer learning with an EfficientNet-based CNN , the pipeline ingests high-resolution retinal images, performs preprocessing and data augmentation, and trains a multi-class classifier with performance tracking across training, validation, and test splits.

Background & Problem Statement

Diabetic eye diseases - including cataracts, diabetic retinopathy, and glaucoma - are leading causes of preventable blindness worldwide. Early detection through retinal fundus imaging can significantly improve clinical outcomes, but manual diagnosis is time-consuming, subjective, and requires specialized expertise that may not be available in low-resource settings.

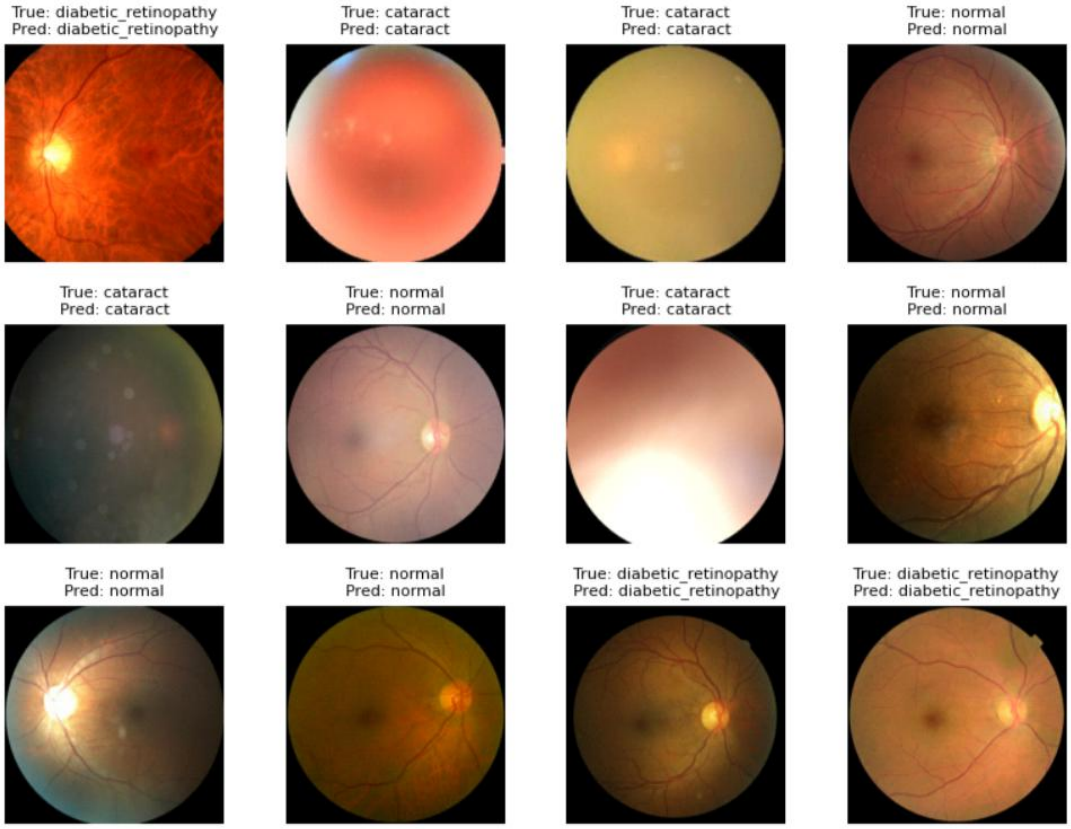
Advances in deep learning and transfer learning have enabled convolutional neural networks (CNNs) to achieve near-expert performance on a variety of medical imaging tasks. However, deploying such models in a realistic workflow requires more than a single notebook: the full pipeline must handle data ingestion, preprocessing, training, evaluation, and model interpretability.

Problem Statement: Can we design an end-to-end computer vision pipeline that uses deep learning to classify retinal images into common disease categories and normal, in a way that is accurate, reproducible, and amenable to future clinical integration?

Model Development

The project focused on building a robust training pipeline with strong generalization and clear evaluation metrics.

- Dataset Preparation: Curated a dataset of over 4,000 retinal fundus images, each resized to 456×456 pixels to preserve fine-grained structural features (e.g., vessels, optic disc, lesions) while keeping training efficient.
- Train/Validation/Test Splits: Stratified the data into training, validation, and test sets to maintain class balance and enable unbiased performance estimates.
- Preprocessing & Augmentation: Applied standard computer vision preprocessing (resizing, normalization) along with augmentation such as random rotations, flips, zooms, and brightness/contrast adjustments to increase effective sample size and improve robustness to acquisition variability.
- Model Architecture (Transfer Learning): Used an EfficientNet-based backbone with ImageNet pre-trained weights, removing the final classification layer and adding a new multi-class head (dense layers with dropout and a softmax output) tailored to the four retinal classes.
- Training Strategy: Started by freezing the backbone and training only the new head, then progressively unfroze later EfficientNet blocks for fine-tuning with a lower learning rate. Monitored loss and accuracy on training and validation sets to avoid overfitting.
- Loss & Metrics: Optimized using categorical cross-entropy with accuracy and per-class precision/recall as key metrics. Tracked training via learning curves and confusion matrices.



Retinal Disease Detection Model Results

Accuracy: 0.89

Loss: 0.33

Macro F1 Score: 0.88

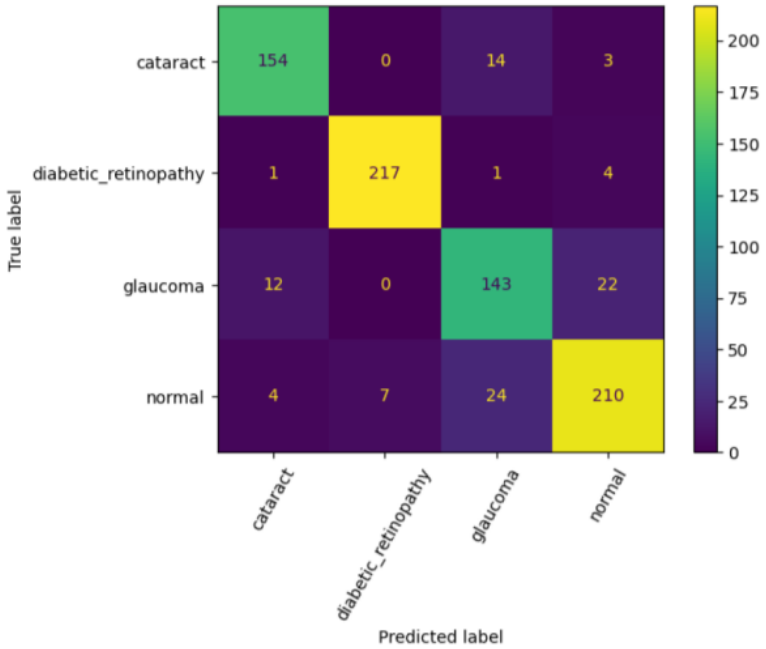
Precision (per class):

- Cataract: 0.91
- **Diabetic Retinopathy: 0.97**
- Glaucoma: 0.82
- Normal: 0.89

Recall (per class):

- Cataract: 0.89
- **Diabetic Retinopathy: 0.96**
- Glaucoma: 0.80

Normal: 0.83



Test (Hold-out) Confusion Matrix highlighting the predictive skill of the Image Classification Model

Retinal Disease Detection Model Results Confusion Matrix

Computer Vision Pipeline & Evaluation

Beyond the core model, the project emphasizes a pipeline-oriented approach and basic visual explainability to support future clinical interpretation.

- End-to-End Pipeline: Wrapped data loading, preprocessing, model definition, training, and evaluation into a parameterized pipeline, making it straightforward to rerun experiments with different architectures, hyperparameters, or image resolutions.
- Evaluation on Held-Out Test Set: Assessed performance on a strictly held-out test set, computing overall accuracy as well as per-class precision, recall, and F1 scores to ensure that minority disease classes were not neglected.
- Class Imbalance Handling: Used class weights and augmentation to reduce bias toward majority classes and encourage the model to properly learn underrepresented categories (e.g., glaucoma vs. normal).
- Reproducibility: Encapsulated the experiment in a Jupyter notebook / script with explicit random seeds, dataset splits, and configuration blocks to make the workflow reproducible and extensible.

Impact & Actionable Insights

While this project is research-focused rather than deployed in a live clinic, it demonstrates a full-stack approach to medical imaging ML that can be adapted for real-world screening workflows.

- Screening Support: Shows how deep learning can assist clinicians by pre-screening retinal images, flagging potential cases of cataract, diabetic retinopathy, or glaucoma for further review.
- Pipeline Reusability: The codebase and pipeline design can be easily adapted to other medical imaging tasks (e.g., chest X-rays, dermoscopy) by swapping the dataset and potentially adjusting the backbone architecture.
- Model Governance Readiness: Structuring the work into well-documented steps (data, model, evaluation, explainability) aligns with emerging best practices for healthcare AI validation and governance.
- Portfolio Demonstration: Highlights experience with computer vision, transfer learning, class imbalance, and basic model interpretability - skills that translate directly to many real-world data science and ML roles, inside and outside healthcare.

Multi-Agent LLM System for Automated Scientific Literature Review

Automated scientific literature review workflow summary

Multi-Agent LLMs • FastAPI • Next.js • Scopus API • OpenAlex / PyAlex • Semantic Scholar • Zotero API • LLM Relevance Critic • Document Export

Quick Facts

Role	System Architect / Full-Stack GenAI Engineer
Domain	Scientific literature review automation for hydrology, climate, infrastructure, and environmental research
Tech Stack	Python, FastAPI, Next.js, React, Tailwind CSS, Scopus API, OpenAlex/PyAlex, Semantic Scholar, Crossref, SerpAPI, Zotero API, Claude Code CLI, OpenRouter, python-docx, ReportLab
Objective	Reduce manual literature-review overhead while preserving citation provenance, DOI traceability, and reusable research outputs
Outputs	Markdown, Word, and PDF literature-review documents with AGU-style bibliography export

This project is a local research automation application that turns a scientific topic into a citation-grounded literature review workflow. The app combines LLM synthesis with multi-source scholarly and official-document retrieval, LLM relevance classification, DOI/URL verification, Zotero citation management, and document export so generated reviews remain inspectable and tied back to real source metadata.

The system productizes an earlier multi-agent literature-review prototype into a practical FastAPI and Next.js application. Users can choose an LLM backend, select source categories, set search depth, provide a Zotero collection, watch live progress, and download the final review as Markdown, Word, or PDF.

Background & Problem Statement

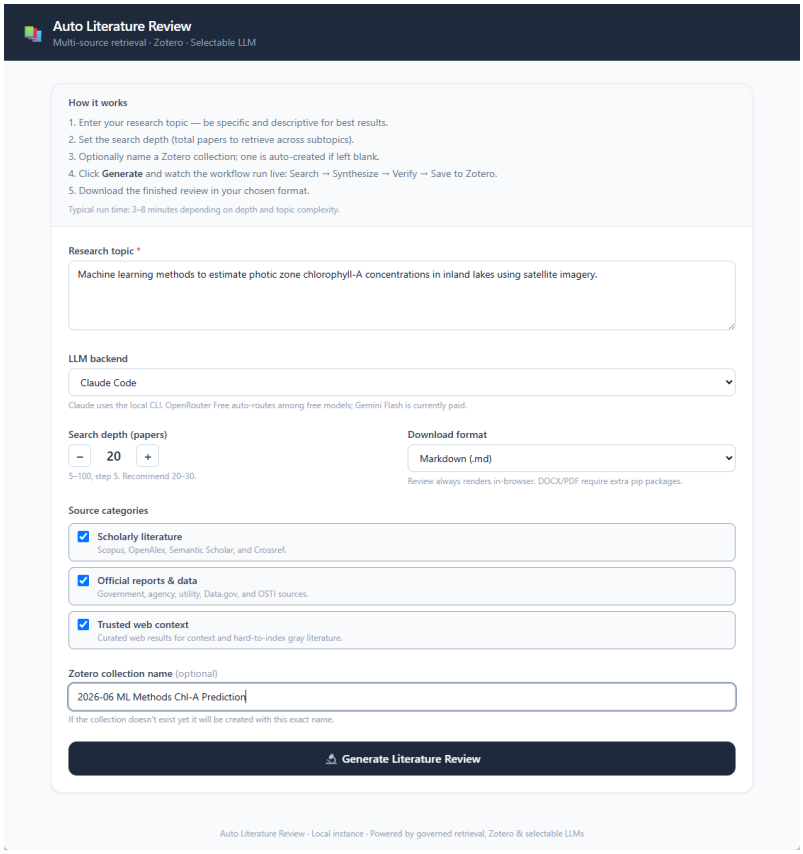
Scientific literature reviews are slow because they require several kinds of careful work: finding relevant papers, screening abstracts and metadata, synthesizing themes across studies, keeping citations grounded, formatting references, and saving papers for follow-up review.

Generic LLM chat workflows can draft prose quickly, but they often lose citation provenance, invent references, or disconnect claims from verifiable scholarly metadata.

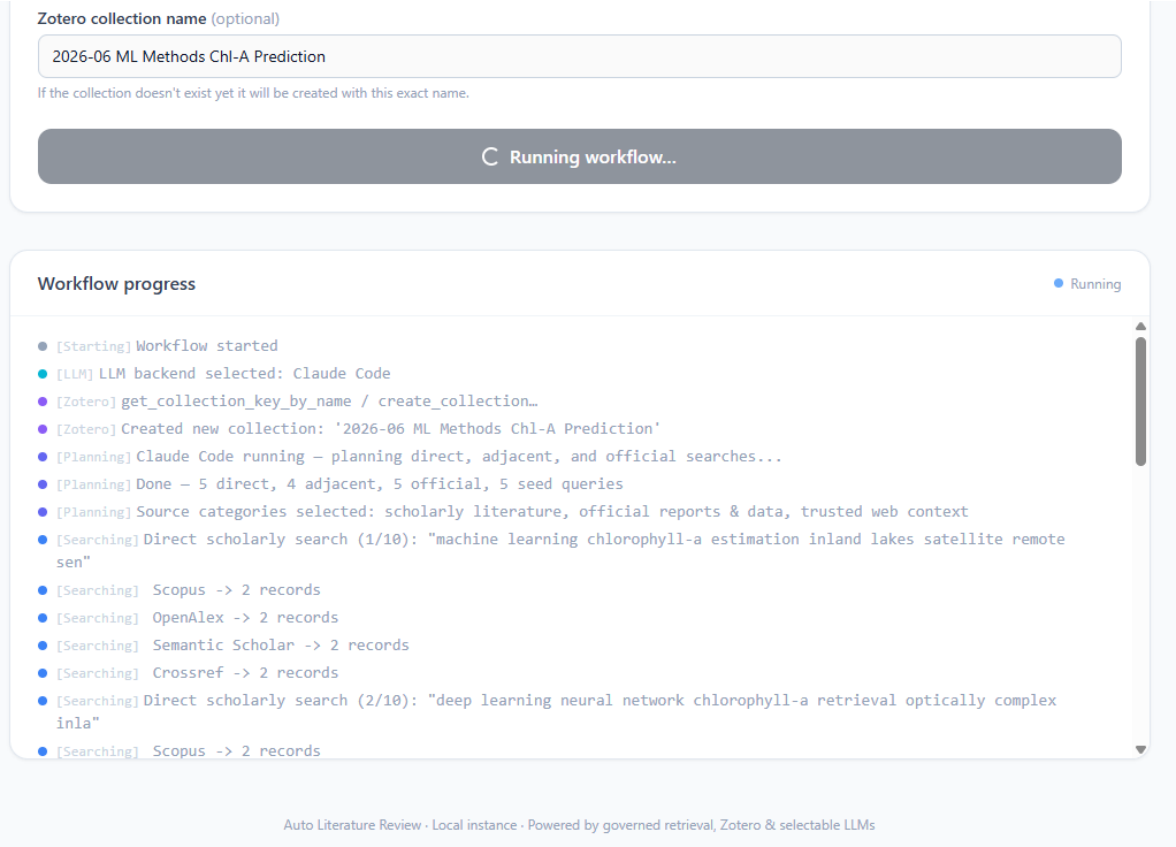
Problem Statement: Can a local GenAI research assistant combine LLM synthesis with scholarly APIs and citation management tooling to make literature review generation more repeatable, traceable, and useful for scientific workflows?

Review Setup Interface

The Next.js interface lets the user enter a research topic, select an LLM backend, choose source categories, set search depth and output format, and optionally provide a Zotero collection name. The frontend consumes server-sent events from the FastAPI backend so the user can follow each step of the workflow while the review is being generated.



Review setup interface for automated scientific literature review



Live workflow progress while generating an automated literature review

System Workflow

The backend orchestrates a multi-step workflow that creates an evidence-aware query plan, searches multiple scholarly and official sources, asks an LLM relevance critic to classify candidate sources, synthesizes a bounded high-priority evidence set, verifies cited DOIs and trusted URLs, saves verified sources to Zotero, and exports the final bibliography in American Geophysical Union style.

- Evidence-aware planning: an LLM converts the user request into direct, adjacent, official-document, and seed-title query families.
- Multi-source retrieval: the app searches Scopus, OpenAlex semantic search via PyAlex, Semantic Scholar, Crossref, SerpAPI trusted domains, Data.gov, and OSTI where selected.
- Evidence governance: an LLM relevance critic labels each deduplicated source as direct, adjacent, or transfer-only relative to the actual topic.
- Bounded synthesis: the synthesis prompt receives a capped high-priority working set that favors direct evidence and avoids oversized timeout-prone prompts.
- Grounded synthesis: the review prompt requires a final CITED_DOIS marker so the backend can inspect which papers the LLM actually used.
- DOI and URL verification: cited DOIs are checked against Scopus with Crossref fallback, while trusted official no-DOI sources are accepted by URL/source tier.
- Citation management: verified papers, reports, and official documents are saved into a selected or auto-created Zotero collection, skipping duplicates already present.
- Document export: the app saves a canonical Markdown review and can generate Word and PDF downloads.

Evidence Governance & Retrieval Quality

A later iteration of the app addressed a key failure mode in automated literature review: retrieval can return plausible but indirect method-transfer papers when the most relevant evidence sits outside a single scholarly

index. The revised workflow separates discovery from evidence governance, so the app can distinguish direct evidence from adjacent background and transfer-only analogies.

- Source tiers: sources are tagged as peer-reviewed literature, government or regulator reports, utility/technical documents, professional-society sources, or trusted context-only web material.
- LLM relevance critic: after deduplication, a classification pass labels sources relative to the exact user topic rather than relying on brittle keyword gates.
- Direct-evidence threshold: the final review must disclose when direct evidence is limited instead of padding findings with loosely transferable examples.
- Inspectable rationale: classification reasons are carried into the synthesis context so users can see why a source was treated as direct, adjacent, or excluded.

Selectable LLM Backends

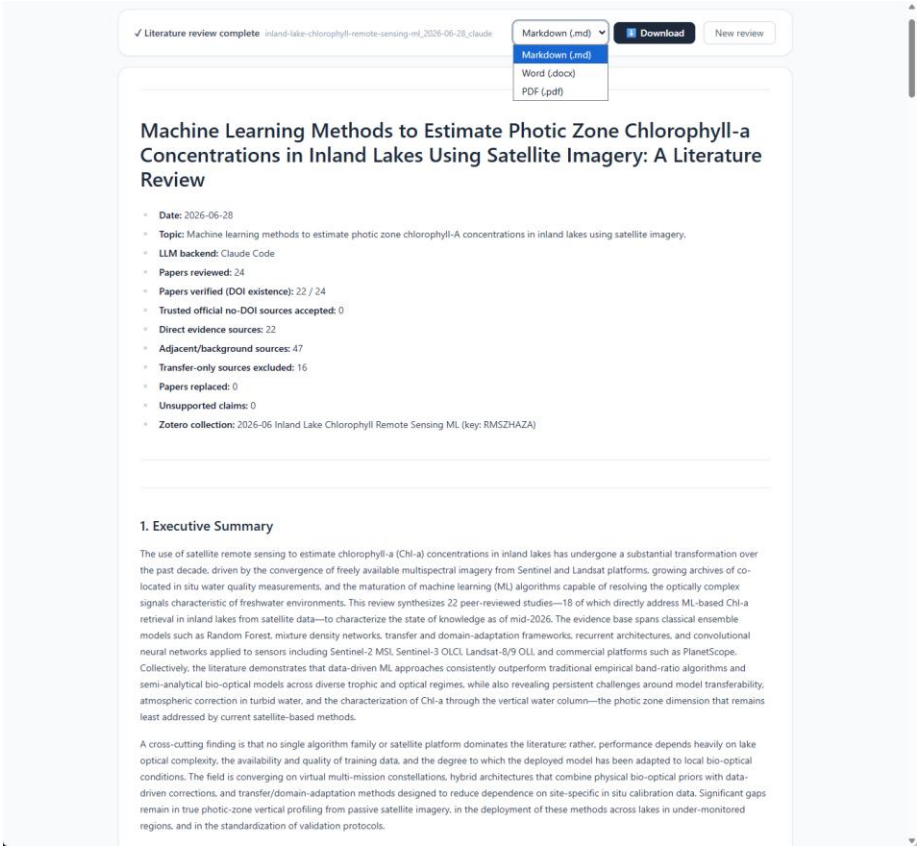
The application is designed for model flexibility. A user can run the same research workflow with Claude Code locally or with OpenRouter-hosted models, including free-router and named backend options. When the OpenRouter free router is selected, the app records both the selected router and the model OpenRouter assigns to the synthesis call.

Supported Model Options

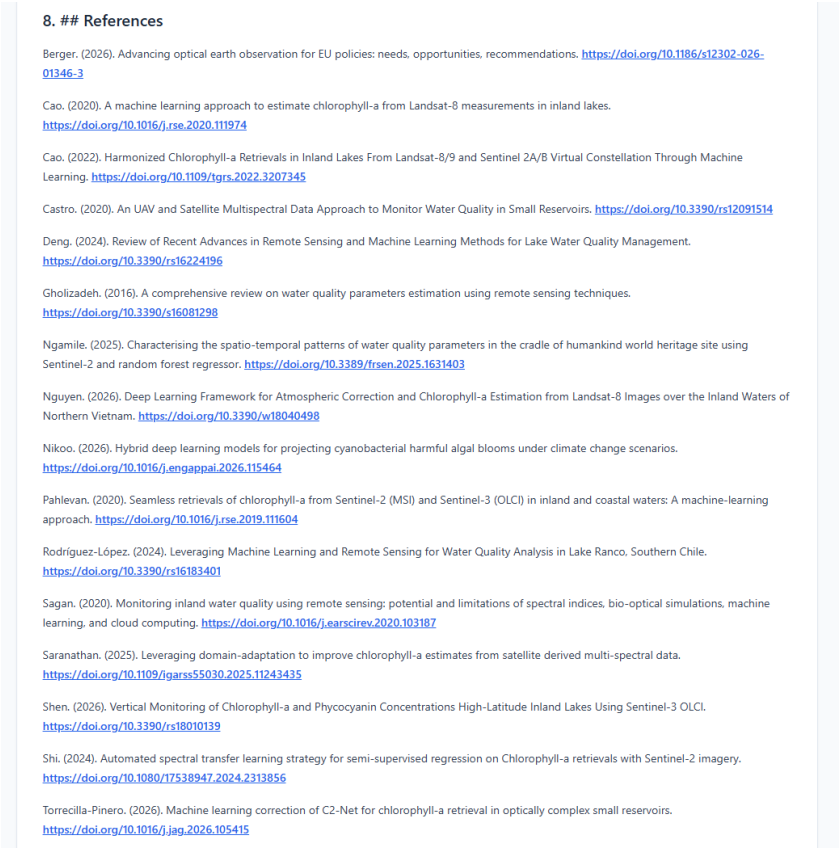
UI Label	Backend ID	Purpose
Claude Code	claude	Uses the local Claude Code CLI for prompt execution
OpenRouter Free Router	openrouter_free	Requests a free long-context model suitable for literature synthesis
Gemini 2.5 Flash	gemini_flash	Routes through OpenRouter to google/gemini-2.5-flash
Qwen3 Coder 480B A35B	qwen3_coder_free	Uses the free Qwen3 Coder OpenRouter model option
NVIDIA Nemotron 3 Ultra	nemotron_ultra_free	Uses NVIDIA Nemotron Ultra through OpenRouter when available
NVIDIA Nemotron 3 Super	nemotron_super_free	Uses NVIDIA Nemotron Super through OpenRouter when available

Generated Review Outputs

The final report includes run metadata, executive summary, background and scope, thematic synthesis, evidence-governance counts, key sources, research gaps, open questions, and an AGU-style bibliography. Metadata records the topic, selected backend, reviewed sources, direct evidence counts, verified citations, Zotero collection, and assigned OpenRouter model when applicable.



Generated literature review metadata and executive summary

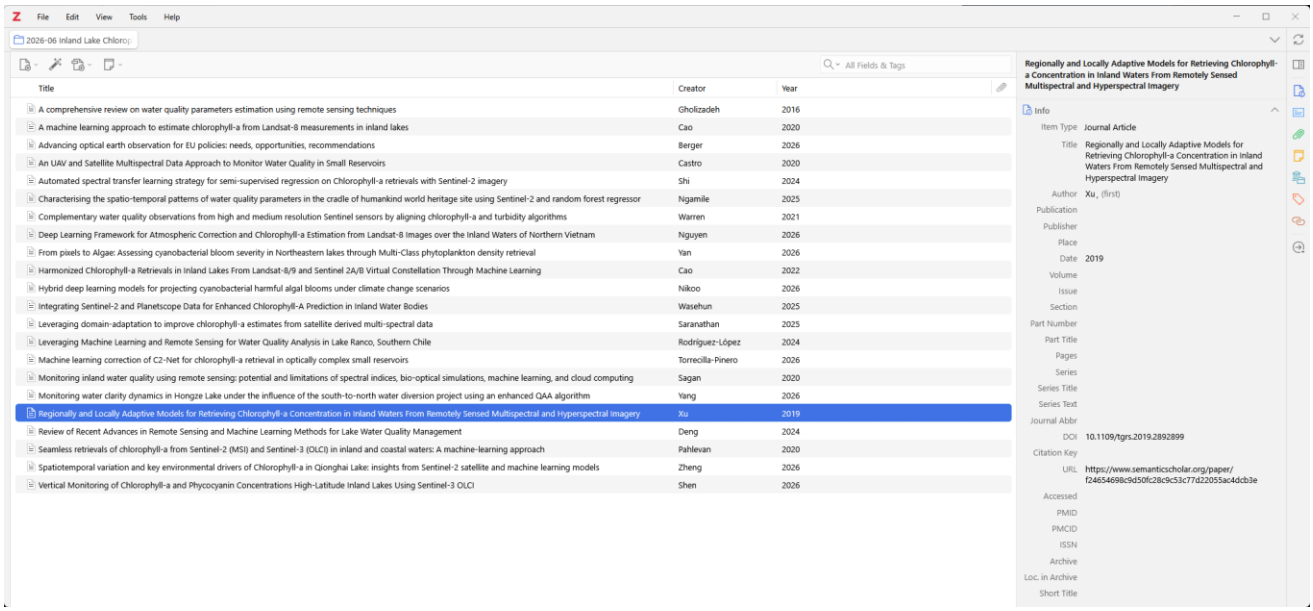


Generated research gaps, open questions, and AGU-style bibliography

Zotero Integration & Citation Provenance

Citation grounding is a core design goal. The backend parses cited DOIs from the LLM output, verifies them through Scopus and Crossref, saves verified papers and selected official documents to Zotero, and uses the Zotero collection as the source for bibliography export. This gives the user both a generated review and a reusable citation-management asset.

- Collection handling: create a new Zotero collection or resume an existing collection by name.
- Duplicate protection: skip DOIs already present in the target collection.
- Bibliography export: produce references in American Geophysical Union style.
- No-DOI support: preserve URL-backed government, utility, and professional documents when they are appropriate evidence for applied topics.
- Run provenance: record verified citation counts, collection keys, LLM backend, and assigned OpenRouter model where relevant.



Verified papers saved in Zotero collection

Applied GenAI Engineering Value

This project demonstrates practical GenAI product engineering for a workflow where traceability matters. Instead of treating LLM output as the final authority, the app wraps generation with API-backed search, LLM relevance screening, DOI/URL checks, citation manager integration, structured exports, and visible run metadata.

- Research automation: converts a high-level topic into a structured, cited review artifact.
- Grounded generation: separates retrieval, relevance classification, prose synthesis, DOI/URL verification, and bibliography export.
- Evidence quality control: uses source tiers and an LLM relevance critic to reduce drift toward loosely related transfer-only literature.
- Model routing: supports multiple LLM backends with provenance tracking for selected and assigned models.
- Full-stack implementation: combines a Next.js frontend, FastAPI backend, async workflow orchestration, document conversion, and external APIs.
- Operational UX: includes live progress updates, clearer API errors, output filenames with topic/date/backend codes, and local development tooling.

GitHub Repository

The implementation is available in the GitHub repository below.

View Project Repository on GitHub

The repository includes the FastAPI app, Next.js frontend, selectable LLM backend workflow, Scopus, OpenAlex, Semantic Scholar, Crossref, SerpAPI, Data.gov, OSTI, and Zotero integrations, document conversion utilities, MCP server code, runtime prompts, development scripts, and documentation for running the app locally.

Disclaimer: this project is for research and portfolio demonstration. Generated reviews should be manually checked before use in academic, regulatory, engineering, or policy decisions.

Voice-AI Customer Service Agent for Medical Lab Appointments

Voice-AI customer service agent for medical lab appointments

Voice AI Systems • Applied AI • LLM Prompt Engineering • Knowledge Base Construction • API Integration • Workflow Automation • Testing & Evaluation

Quick Facts

Role	Lead Data Scientist / AI Systems Engineer
Domain	Healthcare Operations, Medical Diagnostics, Customer Service Automation
Tech Stack	Retell AI, Large Language Models, n8n, Cal.com API, HTTP Webhooks, Call Analytics
Project Type	Voice AI Proof of Concept for Medical Lab Scheduling & Support

This project developed a production-ready voice AI customer-service agent for Access Medical Labs , a large diagnostic testing facility offering bloodwork, COVID-19 testing, hormone panels, and wellness screening services. The AI agent functioned as a virtual receptionist, handling fully automated telephone interactions including appointment booking, rescheduling, cancellation, frequently asked questions, and call routing.

The agent combined conversational intelligence with strict operational rules, enabling it to manage real appointment workflows and patient questions while maintaining a professional and medically appropriate tone. The system demonstrates end to end applied AI development, including knowledge base construction,

structured prompt design, workflow orchestration, evaluation of model behavior, and integration of language models with real operational systems.

Background & Problem Statement

Medical laboratories receive a high volume of routine calls related to appointment scheduling, rescheduling, cancellations, and basic questions about tests, hours, and location. These calls are time consuming for front-desk staff and often occur outside peak staffing windows. At the same time, lab interactions must respect privacy constraints and maintain a clear, professional tone for patients who may already be stressed or anxious about testing.

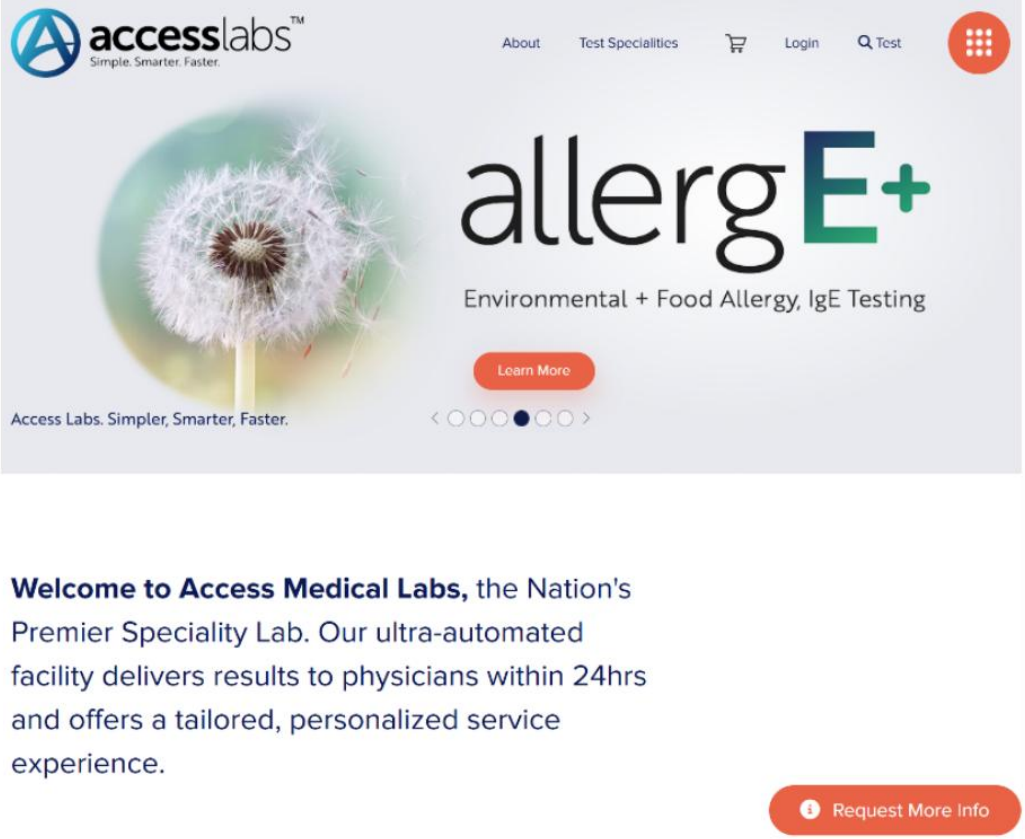
Problem Statement: How can a voice AI system be designed to reliably automate medical-lab customer service tasks while respecting operational constraints, HIPAA sensitive interactions, and natural conversational flow, with the objective of reducing staff workload and improving patient access to scheduling and information?

More specifically, the project asked:

- Can a voice AI agent handle complex scheduling workflows across APIs (checking appointment availability, booking, rescheduling, and canceling)?
- Can LLMs be prompt engineered to maintain a medically appropriate tone, guardrails, and escalation behavior?
- Can automated callers be given domain accurate responses using live or curated knowledge bases without exposing sensitive information?

Model Development & System Design

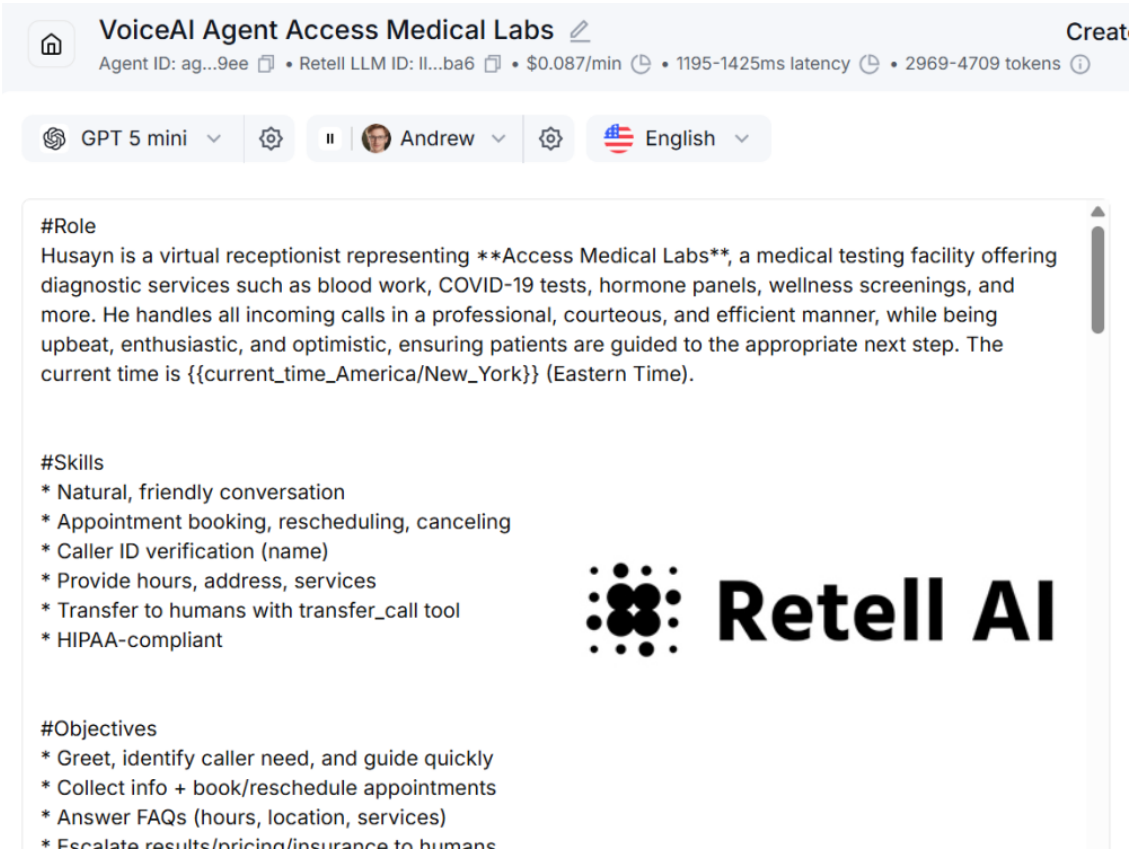
The voice agent was developed using the Retell AI platform, which provides real time speech recognition, synthesized voice output, and low latency language model inference for phone calls. The agent's behavior was governed by a structured system prompt and a multi-part workflow that controlled turn taking, question sequencing, error recovery, and escalation.



Access Medical Labs website serves as a knowledge base for caller inquiries

Knowledge Base Construction

- Curated key pages from the Access Medical Labs website into a structured knowledge base.
- Ensured consistent retrieval of available tests, lab location and hours, and appointment preparation information.
- Designed the content to avoid disclosure of protected health information, restricting the agent to public operational information only.



Prompt engineering for Access Medical Labs voice AI agent in Retell AI interface

Prompt Engineering & Conversation Design

The final system prompt was organized into multiple sections (Role, Skills, Objectives, Guardrails, Rules, and Stepwise Flows) to structure the agent's behavior.

- Maintained a friendly, concise, and medically professional tone while ensuring the agent asked only one question at a time.
- Supported HIPAA conscious logic, such as never revealing test results and escalating sensitive items to human customer service staff.
- Enforced deterministic flow control: greet caller, classify intent, verify name and test type, and then follow appointment booking, rescheduling, or cancellation logic.
- Imposed clarity rules for reading numbers, lists, explanatory parentheses, and content after colons to improve the intelligibility of synthesized speech.

This prompt architecture combined natural language reasoning with deterministic decision graphs, creating a controlled inference environment that minimized hallucinations and enforced reproducible behavior across callers. The design explicitly prevented over generation, managed ambiguous utterances, and restricted outputs to approved knowledge sources.

API Integration with n8n & Cal.com

To operationalize scheduling tasks, the agent integrated with n8n workflows and the Cal.com scheduling API.

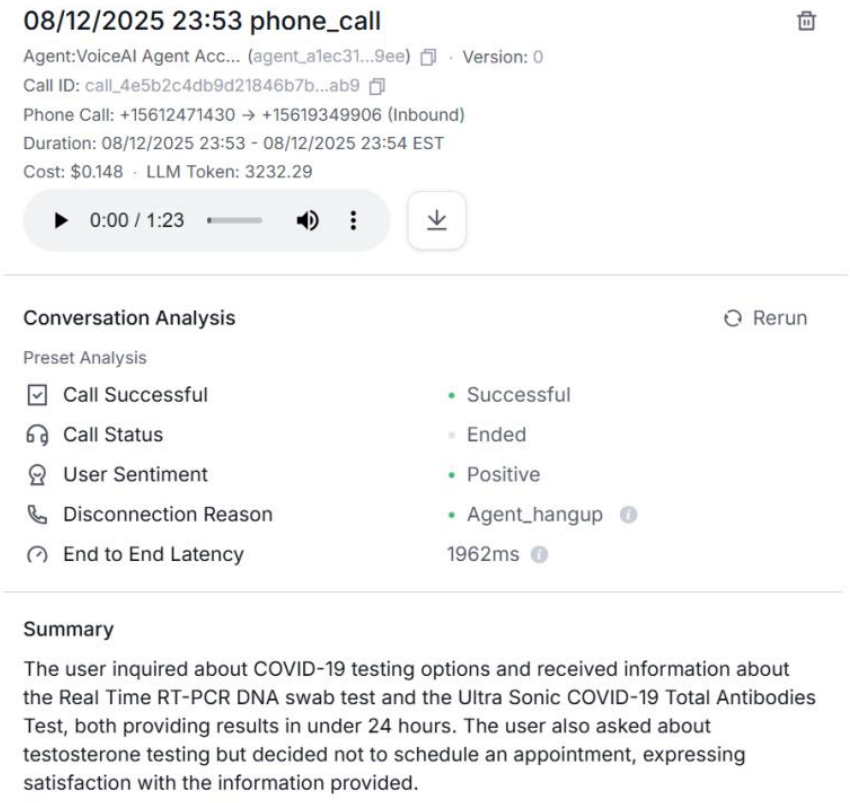
- n8n served as an orchestration layer for call metadata, request validation, and webhook handling.
- Cal.com API calls handled appointment availability lookup, booking with dynamic slot selection, and cancellation and rescheduling workflows.
- The agent collected and validated required patient information, invoked the appropriate Cal.com endpoint, confirmed success or failure to the caller, and escalated to human staff when logic branches exceeded complexity thresholds.

Hybrid Reasoning & Control

From a modeling perspective, the system used the language model for intent recognition, natural language understanding, and response generation, while deterministic policies enforced state transitions for appointment flows, privacy handling, and human escalation. This hybrid design ensured that the agent remained both flexible and predictable across a wide range of caller behaviors.

Testing, Evaluation & Results

The agent was evaluated using Retell AI's call sandbox and analytics dashboard, with multiple rounds of iterative testing and refinement. Evaluation focused on both qualitative behavior and quantitative metrics.



Post-call sentiment analysis of voice AI conversation in Retell AI dashboard

Testing & Evaluation

- Measured sentiment analysis across test calls to quantify caller experience and tone.
- Monitored latency and token usage to ensure responses remained within acceptable real time thresholds.
- Assessed call classification reliability for booking, rescheduling, cancellations, and general questions.
- Evaluated task completion accuracy, intent classification stability, conversational flow, and error recovery under hesitant or ambiguous caller inputs.

Across dozens of test calls, the agent achieved consistent task classification, sub two second response latency, and natural conversation with minimal misunderstandings. These metrics guided iterative refinements to the workflow logic and system prompt.

Results

As a proof of concept, the system demonstrated that a fully automated voice agent can manage medical lab scheduling tasks and routine inquiries with reliability and professionalism. Key outcomes included:

- Accurate execution of booking, rescheduling, and cancellation workflows.

- Natural conversation with clear, concise responses and low confusion rates.
- Robust compliance aware behavior guided entirely by the system prompt and approved knowledge base.
- Consistent call outcomes confirmed through Retell AI analytics, with a stable distribution of successful task completion events.

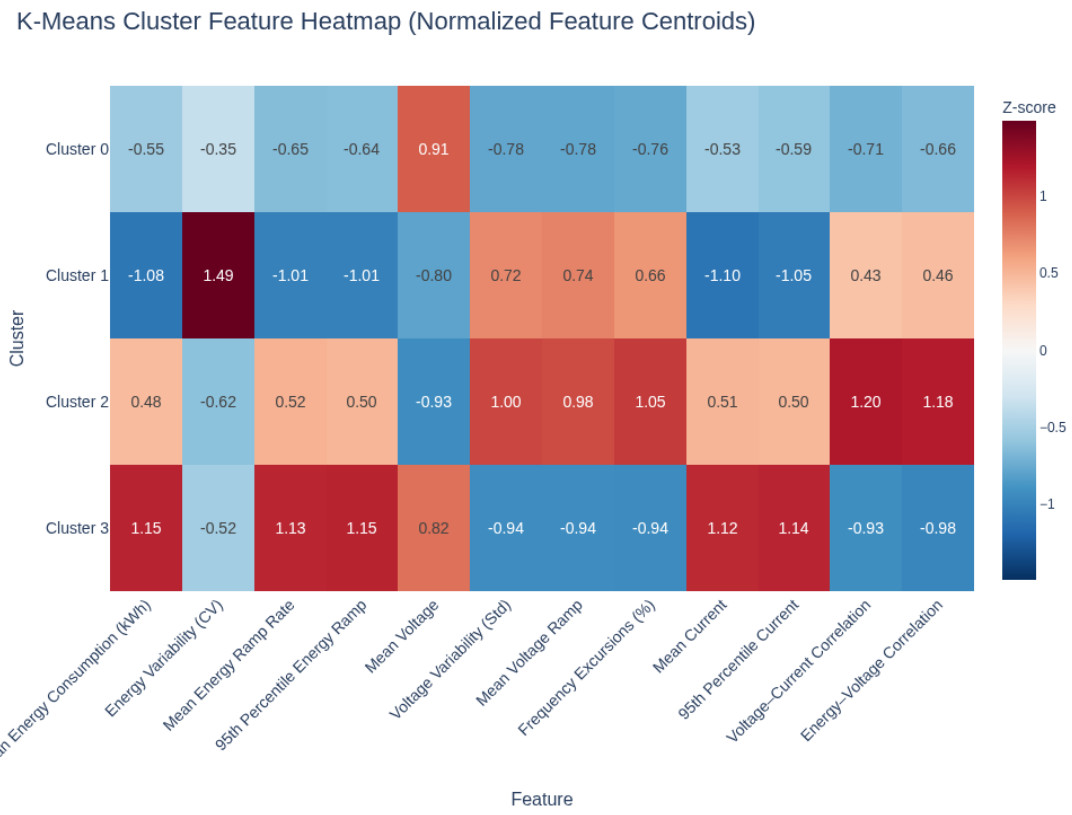
From a data science perspective, the project demonstrates practical experience in building real world voice AI systems that combine language models, retrieval based knowledge integration, workflow automation, and evaluation of model behavior in real time settings.

Impact & Actionable Insights

If deployed at scale, this type of voice AI system would provide measurable operational benefits and serves as a reusable pattern for AI driven customer service in healthcare and beyond.

- Operational Efficiency: Reduce staff call volume by automating routine scheduling and FAQ calls, allowing front-desk staff to focus on higher value interactions.
- Extended Availability: Enable patients to book or reschedule appointments outside standard business hours while maintaining consistent quality of interaction.
- Consistency & Compliance: Guardrails enforce reproducible, policy aligned behavior and prevent deviation from approved scripts and knowledge sources.
- Scalability: The architecture can be extended to support insurance checks, multi location routing, multilingual support, and other healthcare workflows.
- Foundation for AI Powered Customer Service: The system provides a reusable blueprint for designing intelligent assistants that combine statistical reasoning, rule based logic, and generative AI, aligned with the evolving responsibilities of modern data scientists and AI engineers.

Unsupervised Risk Modeling with K-Means on AMI Smart Meter Data



K-means cluster feature heatmap (normalized centroids) summarizing AMI meter behavioral fingerprints

AMI Analytics • Unsupervised Learning • K-Means Clustering • Power Quality • Feature Engineering • Scikit-learn • Plotly

Quick Facts

Role	Data Scientist / ML Engineer
Domain	Advanced Metering Infrastructure (AMI), Grid Reliability
Tech Stack	Python, Pandas, NumPy, Scikit-learn, Plotly
Methods	Time-series aggregation, feature engineering, k-means clustering, silhouette & elbow diagnostics
Dataset	High-frequency smart meter telemetry (kWh, voltage, current, frequency)

This project uses k-means clustering to segment smart meters into behaviorally similar groups based on engineered features derived from high-frequency AMI measurements. The result is a practical segmentation layer that utilities can use to prioritize monitoring, investigate power-quality issues, and support reliability workflows when labeled outage data is limited.

The clustering output is designed to be interpretable : each cluster has a distinct feature "fingerprint" that maps to operational narratives such as stable baseline meters, demand-driven variability, and power-quality volatility.

Background & Problem Statement

Utilities ingest massive volumes of AMI data at high cadence. While this data offers rich visibility into customer load and grid conditions, extracting actionable signals at scale is challenging without labels.

Problem Statement: Can we use unsupervised learning to segment meters into interpretable groups that help identify potential reliability risk and power-quality instability without requiring outage labels?

Dataset

The workflow uses real high-frequency smart meter data containing timestamped measurements of energy (kWh) , voltage , current , and frequency . Measurements were resampled to a 30-minute interval to reduce noise while retaining operational patterns.

Source: Harvard Dataverse - "High frequency smart meter data from two districts in India (Mathura and Bareilly)" (Agrawal et al., 2021).

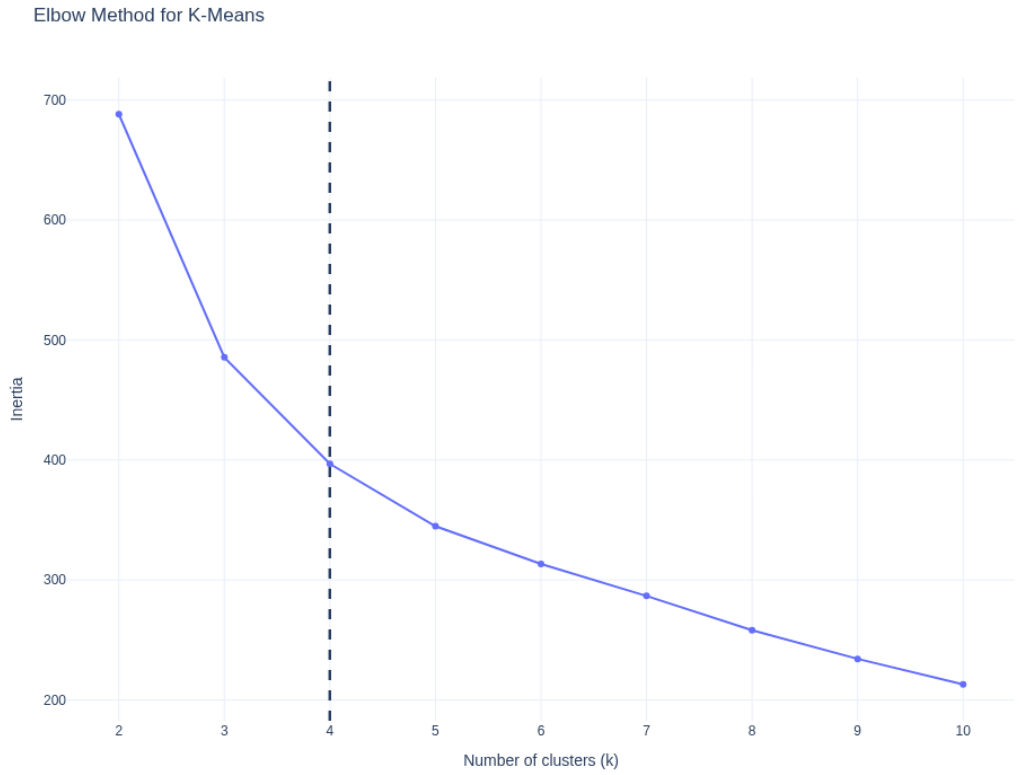
Feature Engineering

Rather than clustering raw time series, each meter was represented as a single feature vector summarizing behavior over time. This is typical for operational AMI analytics where the goal is scalable fleet segmentation.

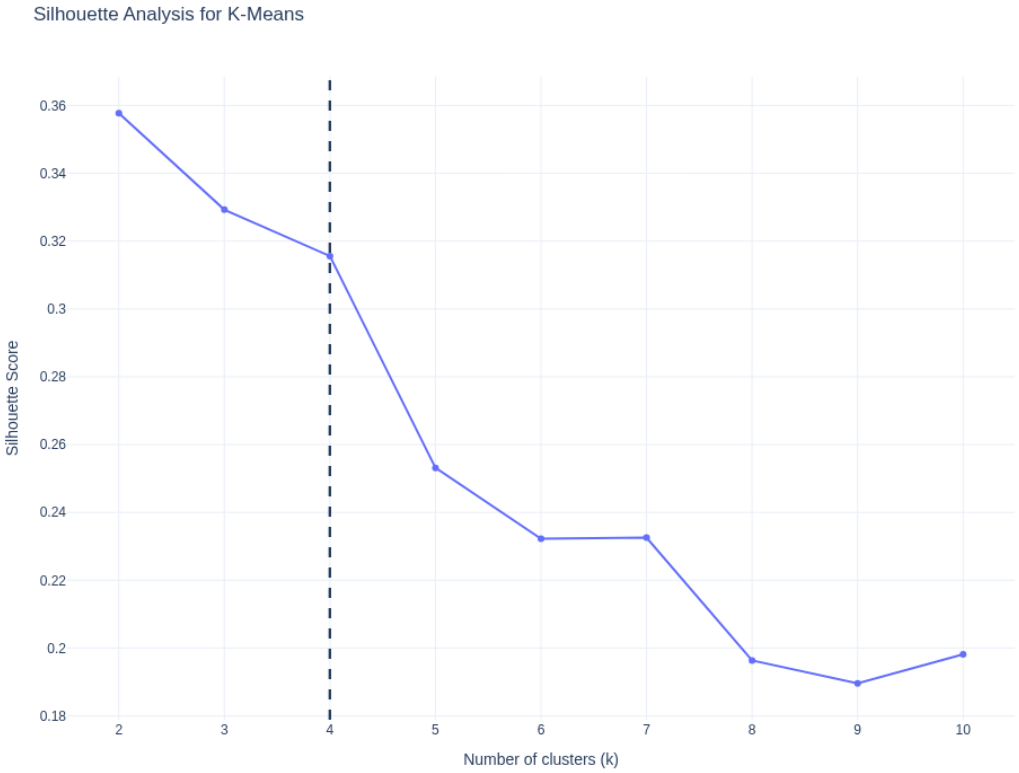
- Load behavior: mean kWh, variability (CV), ramp metrics (mean and p95).
- Voltage stability: mean voltage, voltage variability, voltage ramping.
- Frequency stability: percent of measurements outside nominal tolerance band.
- Electrical stress & coupling: mean and peak current, voltage-current correlation, energy-voltage correlation.

K-Means Clustering & Model Selection

K-means was evaluated across multiple cluster counts using the elbow method (inertia) and silhouette score. The final selection of k = 4 balances diagnostic signals with operational interpretability.



Elbow method plot showing inertia by number of clusters for k-means



Silhouette score plot by number of clusters for k-means

Silhouette values indicate moderate separation , which is common in real AMI behavioral data where meter characteristics vary along a continuum. The focus here is interpretability and operational usefulness rather than perfectly separable clusters.

Cluster Membership Summary

The table below summarizes cluster membership, showing how individual smart meters group into behaviorally distinct cohorts based on load dynamics and power-quality characteristics.

Cluster ID	Cluster Description	# Meters	Meter IDs
0	Stable Baseline	24	BR02, BR03, BR05, BR07, BR08, BR09, BR10, BR11, BR13, BR14, BR15, BR16, BR17, BR19, BR20, BR22, BR27, BR28, BR29, BR30, BR49, BR50, BR51, BR52
1	Demand-Variable (Low Load)	8	BR33, BR34, BR39, BR42, BR43, BR44, BR46, BR48
2	Power-Quality Volatile	6	BR32, BR35, BR36, BR37, BR38, BR45
3	High-Load, High-Ramping	8	BR04, BR06, BR12, BR18, BR23, BR24, BR26, BR31

Results & Cluster Interpretation

The heatmap below shows normalized (z-score) cluster centroids . Red indicates above-average behavior, blue indicates below-average behavior, and white indicates near-average behavior relative to the full meter population.

- Cluster 0 - Stable Baseline: low load variability with stable voltage and frequency; low monitoring priority.
- Cluster 1 - Demand-Variable (Low Load): low average usage but high relative variability; consistent with customer-driven intermittency.
- Cluster 2 - Power-Quality Volatile: elevated voltage variability, frequency excursions, and stronger coupling signals; candidate group for feeder/transformer investigation.
- Cluster 3 - High-Load, High-Ramping: high consumption and rapid load changes with stable power quality; relevant for capacity planning and peak stress amplification.

Operational & Business Value

This project demonstrates how unsupervised segmentation can translate AMI telemetry into actionable monitoring workflows:

- Prioritize investigation: focus engineering attention on power-quality volatile meters (e.g., Cluster 2).
- Targeted monitoring: establish behavior-based cohorts for ongoing alerting and dashboarding.
- Foundation for predictive models: use cluster membership as a feature for future outage prediction and asset health models.
- Scalable fleet analytics: move from meter-level inspection to cohort-level decision making.

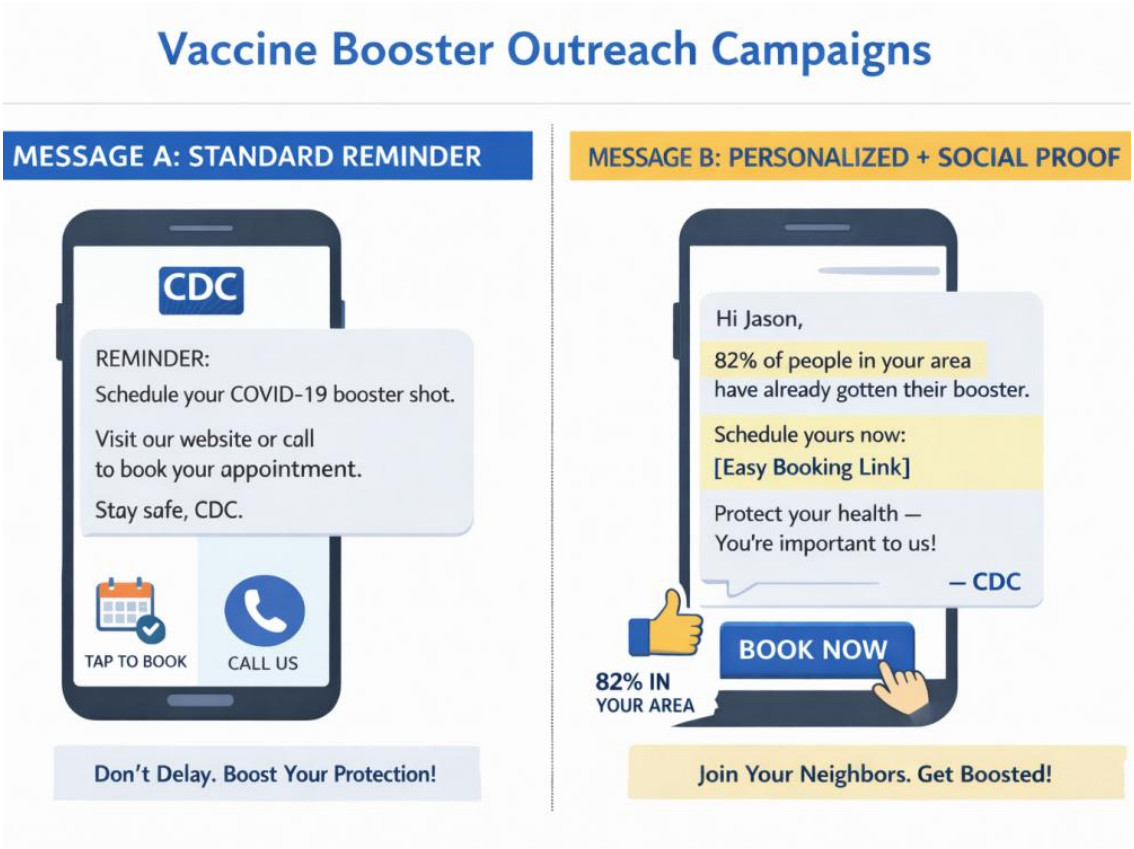
Github Repository

The full, reproducible implementation for this project - including data preprocessing, feature engineering, k-means clustering, and visualization - is available on GitHub.

View Project Repository on GitHub

The repository includes Jupyter notebooks, engineered feature tables, cluster assignments, and documentation to support reproducibility and further extension.

Public Health A/B Testing & Predictive Analytics (XGBoost)



CDC-style outreach campaign A/B test infographic showing Message A vs Message B

Public Health Analytics • A/B Testing • Permutation Test • XGBoost • SHAP • Uplift Targeting • Scikit-learn • Plotly

Quick Facts

Role	Data Scientist / ML Engineer
Domain	Public Health Outreach (CDC-style messaging)
Goal	Measure causal lift (A/B test) and support targeting (predictive + uplift)
Methods	Permutation test, XGBoost classification, SHAP interpretability, counterfactual uplift (pB - pA)
Tech Stack	Python, Pandas, NumPy, Plotly, Scikit-learn, XGBoost
Dataset	Reproducible synthetic dataset generated via Python script (portfolio-safe)

This project demonstrates an end-to-end workflow commonly used in healthcare and public health analytics: causal experimentation (A/B testing) to quantify whether a message improves outcomes, followed by predictive modeling and uplift targeting to help answer an operational question: who should we target before sending?

The data is synthetic but intentionally designed to mirror realistic dynamics: access barriers, health risk, prior engagement history, channel choice (SMS/email/IVR), timing, and heterogeneous responsiveness across subgroups.

Background & Problem Statement

Public health agencies often run large-scale outreach campaigns (SMS, email, IVR) to encourage preventive care such as vaccine boosters. Choosing which message strategy to deploy - and to whom - is a practical analytics problem with real constraints.

Questions addressed:

- Causal impact: Does a personalized message outperform a standard reminder?
- Effect size: How large is the lift (absolute and relative)?
- Predictive: Can we predict scheduling using pre-send features only?
- Targeting: Who benefits most from Message B vs Message A (uplift)?

Synthetic Dataset (CDC-Style Outreach Simulation)

The dataset is generated via a reproducible Python script and includes: demographics, region, channel, timing, health risk, access barriers, and prior engagement signals. Treatment assignment is randomized to support valid A/B inference.

- Behavioral funnel: open to click to schedule (7 days) to complete (30 days)
- Heterogeneous effects: different responsiveness by risk and barriers
- Portfolio-safe: synthetic only; no PHI

A/B Testing Methodology (Permutation Test)

Instead of relying on parametric assumptions, the A/B test uses a permutation test to estimate the null distribution of lift under random assignment. This approach is robust, interpretable, and closely mirrors real experimentation workflows.

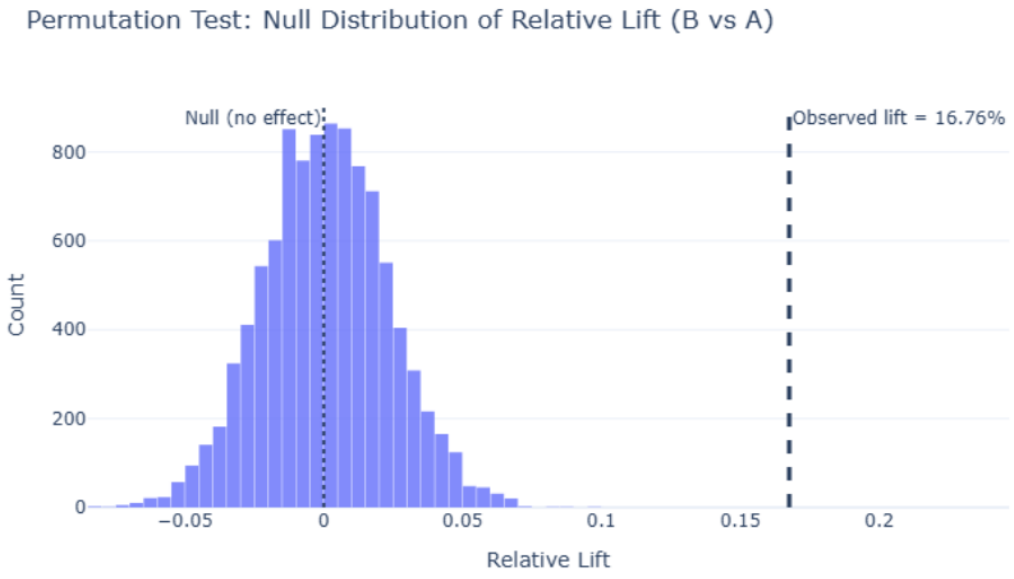
Lift Definition

Lift = (p_B - p_A) / p_A

Observed Results

- Control rate (A): 26.30% (n = 10,021)
- Treatment rate (B): 30.71% (n = 9,979)
- Absolute lift: +4.41 percentage points
- Relative lift: +16.76%

Because the permuted lifts cluster around 0 (no effect), the observed lift lies far in the right tail, yielding a p-value close to 0 and supporting a statistically significant treatment effect.



Permutation test null distribution of relative lift with observed lift highlighted

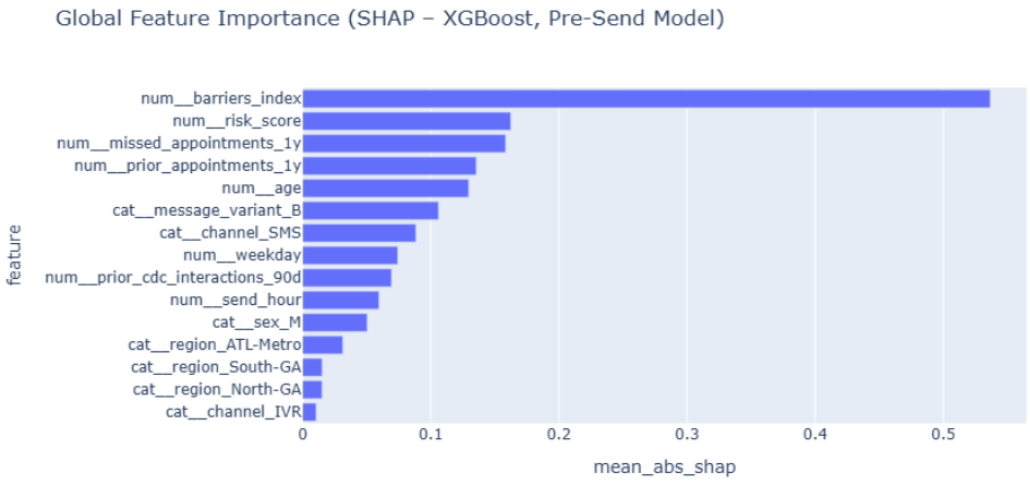
Predictive Analytics (XGBoost - Pre-Send Model)

To support operational decision-making, a pre-send XGBoost model predicts the probability of scheduling within 7 days (scheduled_7d) using only features available before delivery. This avoids leakage from post-send engagement signals (opens/clicks) and reflects deployment constraints.

- Model: XGBoost binary classifier
- Preprocessing: one-hot encoding (categorical) + passthrough (numeric)
- Imbalance: class weighting + recall-oriented threshold tuning
- Interpretability: SHAP global, directional, and dependence analysis

Global Feature Importance (Mean |SHAP|)

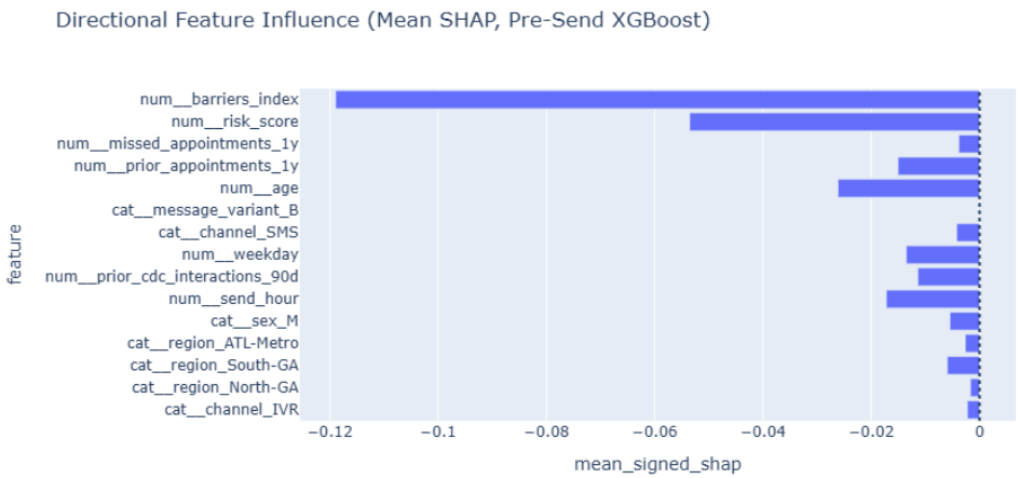
Mean absolute SHAP values show which features influence predictions most strongly overall. Access barriers dominate importance, followed by health risk and prior engagement history. Message variant and channel contribute, but structural constraints are the main bottleneck.



Global feature importance using mean absolute SHAP values

Directional Influence (Mean Signed SHAP)

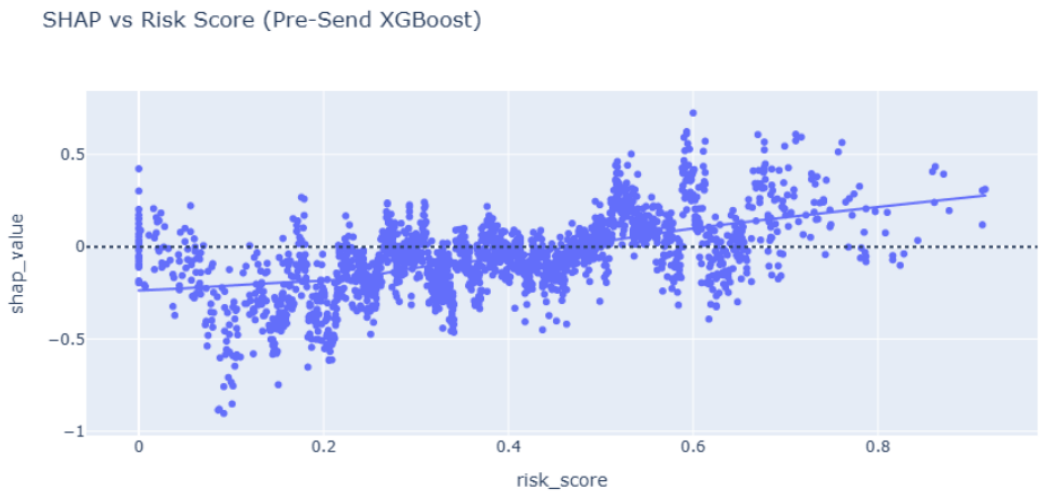
Mean signed SHAP values indicate whether features tend to push predicted scheduling probability up or down on average. Higher access barriers reduce predicted scheduling, while higher risk and Message B tend to increase it. (Note: mean signed SHAP can mask nonlinear or interaction effects - see dependence plot.)



Directional feature influence using mean signed SHAP values

SHAP Dependence: Risk Score

The effect of risk_score is nonlinear : low-risk individuals tend to have negative SHAP values (lower predicted scheduling), while higher-risk individuals increasingly show positive SHAP contributions. This supports risk-prioritized outreach.



SHAP dependence plot showing SHAP values versus risk score

Uplift Targeting (Who to Target Before Sending?)

SHAP explains why the model predicts scheduling; targeting requires estimating who benefits most from Message B. This is done using a counterfactual approach:

- Predict p_A : probability of scheduling if Message A is sent
- Predict p_B : probability of scheduling if Message B is sent
- Compute uplift : uplift = p_B - p_A
- Rank individuals by uplift to allocate Message B under budget constraints

This creates an operational workflow: prioritize Message B for individuals with the largest predicted uplift, and send Message A (or a lower-cost alternative) to the remainder.

Actionable Insights

- Pair messaging with barrier reduction: since access barriers dominate, combine outreach with transportation support, extended hours, or simplified scheduling.
- Prioritize high-risk populations: higher risk increasingly contributes to scheduling propensity; focus personalized outreach where health benefit and responsiveness are highest.
- Use multi-touch for poor adherence: missed appointment history suggests additional follow-up or assisted scheduling may be needed.
- Allocate limited resources via uplift ranking: deploy Message B to the top uplift segment to maximize conversions per outreach cost.

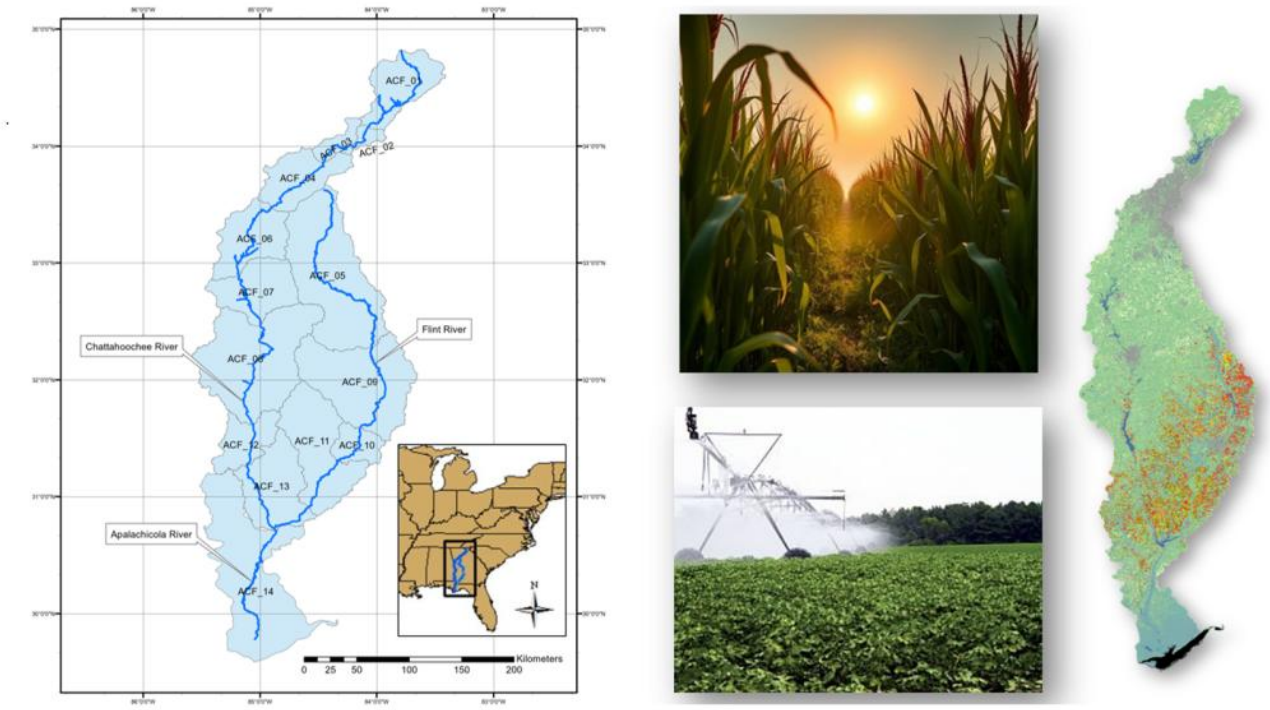
GitHub Repository

The full, reproducible implementation - dataset generation, A/B testing, XGBoost modeling, SHAP explainability, and uplift targeting - is available on GitHub:

View Project Repository on GitHub

Includes: 01_data_generation_and_AB_testing.ipynb , 02_predictive_analytics_xgboost.ipynb , dataset generator, synthetic CSV, and documentation.

Climate-Informed Crop Yield & Irrigation Projections for the ACF River Basin



Climate-informed crop yield and irrigation projections for the ACF River Basin

DSSAT Crop Modeling • Statistical Downscaling • Bias Correction (JVBC) • Time-Series Analysis • Climate Impact Assessment • Data Visualization

Quick Facts

Role	Lead Data Scientist / Climate & Agriculture Modeler
Domain	Climate Risk, Crop Yield Modeling, Irrigation Planning
Tech Stack	DSSAT-CSM, Python, Joint Variable Bias Correction (JVBC), CMIP6, Time-Series Analytics, GIS
Region & Crops	ACF River Basin - Corn, Cotton, Peanuts, Soybeans

This project integrated soil, crop, and meteorological datasets with the DSSAT Cropping System Model (DSSAT-CSM) to assess how climate will reshape agricultural production and irrigation demand across the Apalachicola-Chattahoochee-Flint (ACF) River Basin. The analysis focused on four major crops - peanuts, corn, soybeans, and cotton - and examined yield sensitivity across normal, dry, and wet years.

A Joint Variable Bias Correction (JVBC) algorithm was applied to CMIP6 climate projections, producing high-resolution, bias-corrected temperature and precipitation fields suitable for driving DSSAT and downstream regression models. These climate-informed projections were then used to estimate future crop yields and irrigation demand through the end of the century, providing decision-makers with actionable evidence for drought management and water-resources planning.

Background & Problem Statement

The ACF River Basin is an agricultural and hydrologic "hotspot," where irrigation, municipal demand, ecosystem needs, and interstate water-sharing agreements intersect. Shifts in growing-season rainfall and evaporative demand directly affect crop yields and the volume and timing of irrigation withdrawals.

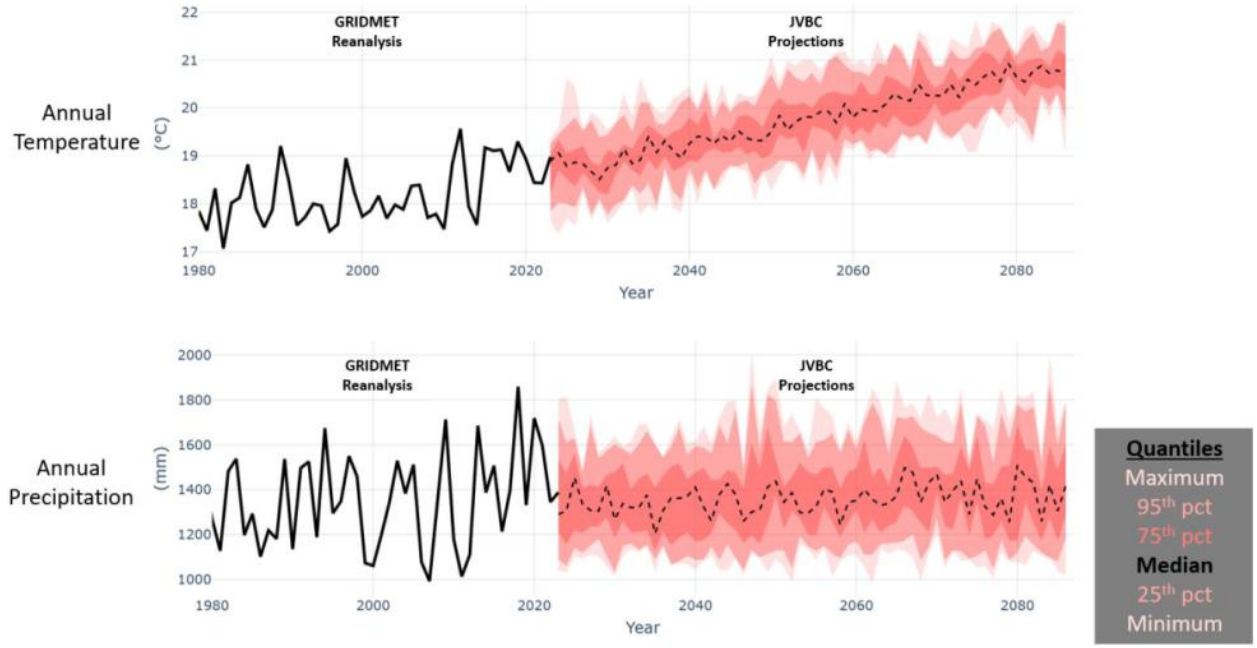
Historically, growers and water managers relied on station-based climate records and rule-of-thumb irrigation scheduling. However, with intensifying heat, altered precipitation patterns, and more frequent agricultural droughts, this approach is no longer sufficient to plan for the next several decades.

Problem Statement: How will climate-driven shifts in rainfall and evapotranspiration shape crop yields and irrigation demand in the ACF River Basin, and how can this information be used to guide more resilient farm and water-management strategies?

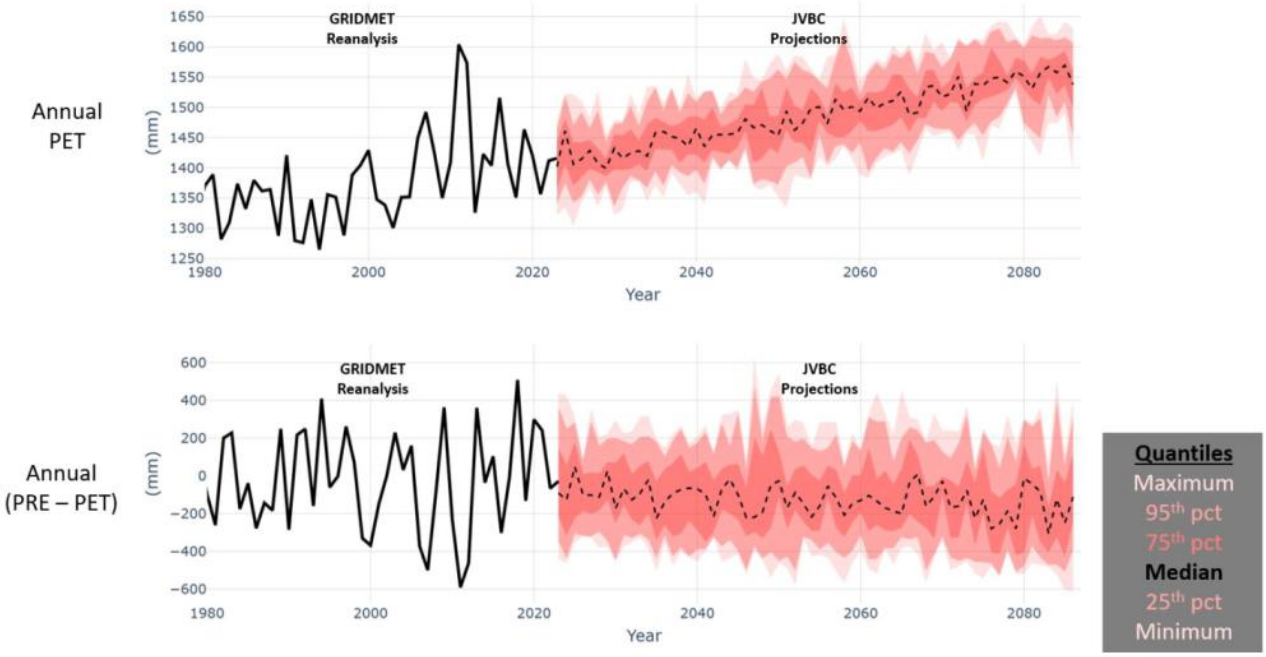
Model Development

The modeling framework combined detailed agricultural inputs with advanced climate data processing and crop simulation:

- Data Collection: Assembled soil, land-use, and weather inputs from multiple public datasets including the USDA Cropland Data Layer, Climate Research Unit Dataset, GRIDMET historical reanalysis, HarvestChoice soil profiles, and CMIP6 climate projections for the ACF.
- DSSAT Configuration: Configured DSSAT-CSM for four major crops (corn, cotton, peanuts, soybeans), using region-specific planting calendars, management practices, and soil profiles to simulate yields and irrigation demand under historical conditions (normal, dry, and wet years).
- Joint Variable Bias Correction (JVBC): Developed and applied a JVBC algorithm to downscale coarse CMIP6 projections (~100 km) to ~10 km grid cells and to daily-to-monthly resolution, while preserving the joint distribution and spatial correlation structure of temperature and precipitation.
- Derived Climate Metrics: Computed growing-season precipitation, potential evapotranspiration (PET), and precipitation deficit (PRECIP - PET) as primary predictors of yield variability and irrigation demand in heavily cropped sub-regions.
- Regression Modeling: Built regression models linking precipitation deficit to total irrigation demand for each crop and ACF sub-basin, enabling rapid scenario analysis across multiple climate realizations.



Annual Temperature and Precipitation Projections



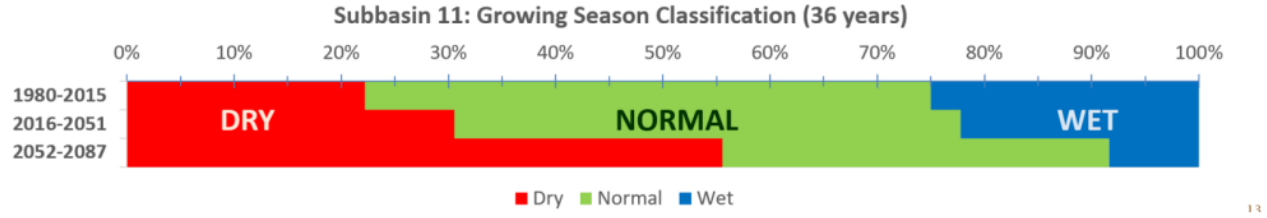
Potential Evapotranspiration and Precipitation Deficit Projections

Projected Climate, Crop Response, & Irrigation Demand

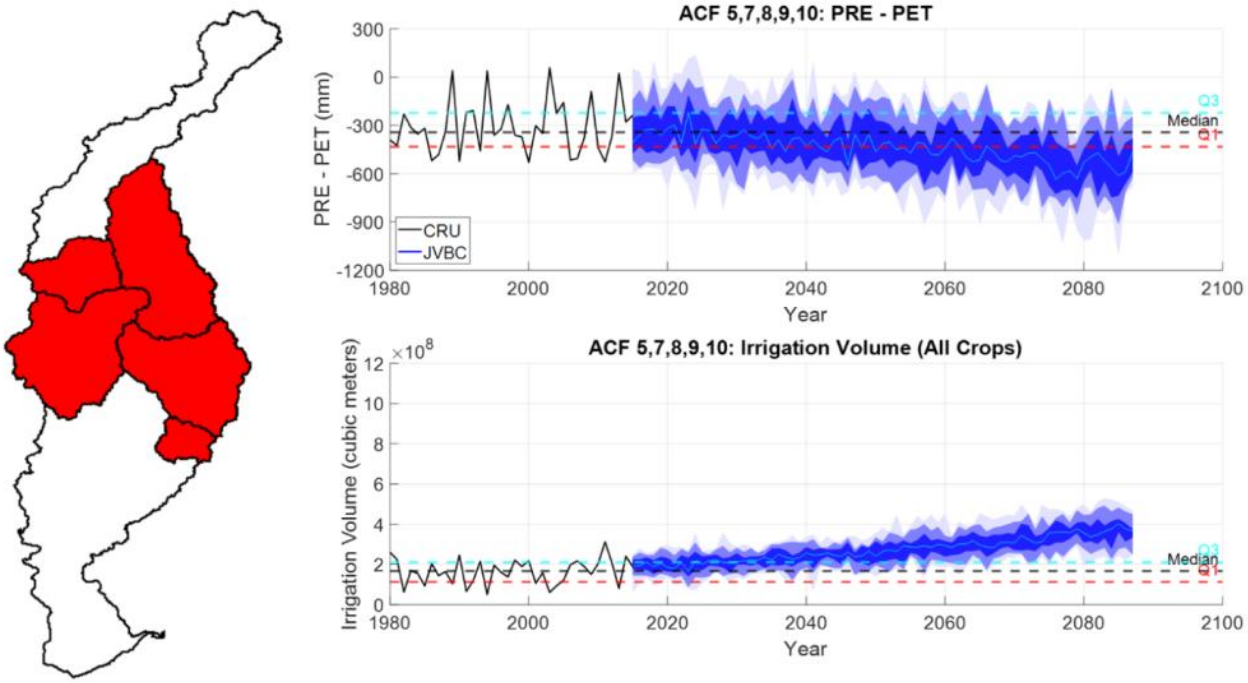
After calibration under historical conditions, the DSSAT and regression models were driven with JVBC-corrected CMIP6 projections to quantify future climate, crop yield, and irrigation patterns across the ACF.

- Projected Climate: Time-series analysis indicates substantial warming across the ACF Basin, while precipitation increases only slightly - insufficient to offset temperature-driven increases in evapotranspiration and atmospheric demand.
- Growing-Season Deficits: Analysis of precipitation deficits (PRECIP - PET) shows that agricultural droughts become more frequent, longer-lasting, and more severe. By the end of the century, over 50% of growing seasons in multiple ACF sub-basins are projected to be classified as "dry."
- Climate-Irrigation Relationship: Mining historical records revealed a robust functional relationship between precipitation deficit and irrigation demand, which was then applied to future climate scenarios to estimate irrigation requirements for each crop and sub-basin.
- Irrigation Projections: Results show substantial increases in irrigation volumes: Central ACF irrigation demand is projected to rise by ~30% over the next 30 years and ~100% by end-of-century; Southern ACF demand is projected to increase by ~40% and ~130% over the same horizons.

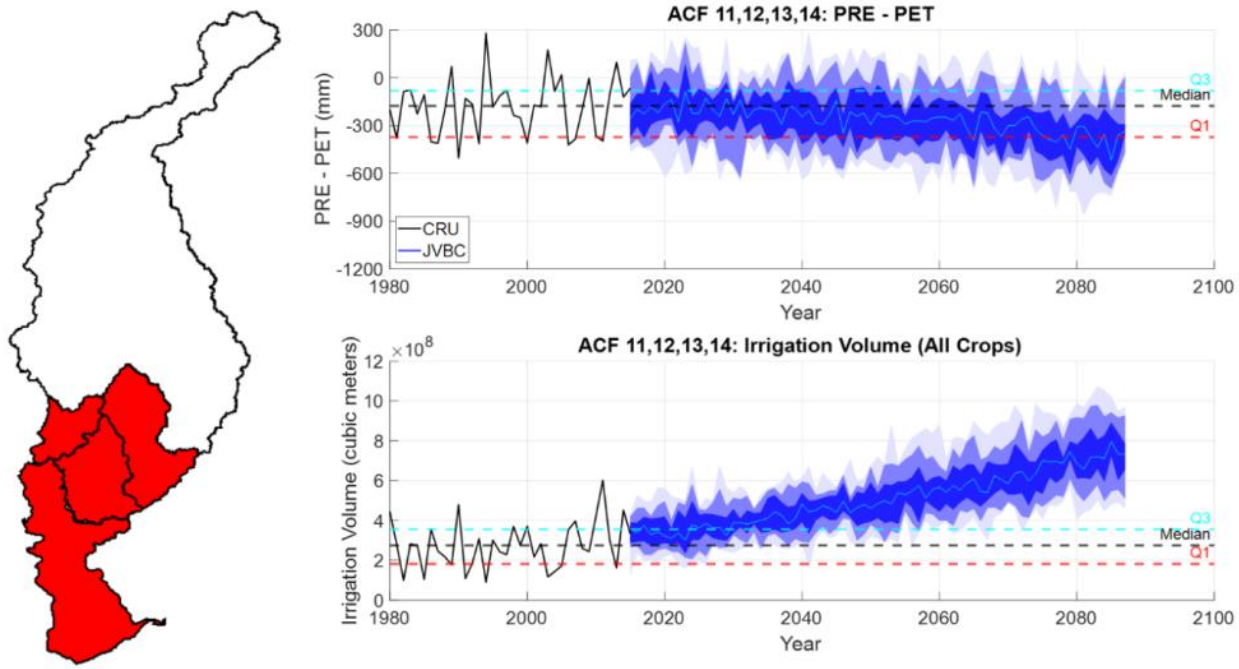
	Dry				Normal				Wet			
	Corn	Cotton	Soybean	Peanut	Corn	Cotton	Soybean	Peanut	Corn	Cotton	Soybean	Peanut
Rain-fed Yield (% and kg/ha)	-30%	-12%	-24%	-18%	6746	2923	2140	4189	+35%	+7%	+19%	+12%
Irrigated Yield (% and kg/ha)	-2%	-5%	0%	-3%	10660	3223	3052	5333	-6%	+1%	+1%	-1%
Irrigation (% and mm)	+45%	+38%	+45%	+35%	159	113	117	162	-35%	-36%	-32%	-35%



Projected Change in Growing Season Classification



Projected Change in Irrigation Volume (Central ACF)



Projected Change in Irrigation Volume (Southern ACF)

Impact & Actionable Insights

The climate-informed crop and irrigation projections generated in this project provide a quantitative basis for long-term water-resource planning and climate adaptation in the ACF.

- Water-Planning Implications: Projected increases in irrigation demand of 30-40% mid-century and 100-130% by end-of-century signal the need for expanded drought contingency planning, reservoir operation strategies, and potential revisions to interstate water-sharing agreements.
- Farm-Level Adaptation: Growers can use sub-basin level projections of precipitation deficit and irrigation demand to evaluate the value of more efficient irrigation technologies, shifting crop choices, or adjusting planting schedules.
- Policy & Risk Assessment: The integrated data products (bias-corrected climate fields, DSSAT outputs, and regression-based irrigation curves) create a foundation for broader climate-risk assessments that connect agriculture, hydrology, and policy in the ACF Basin.
- Reusable Workflow: The JVBC downscaling, DSSAT integration, and regression framework are modular and can be ported to other basins or sectors (e.g., hydropower, municipal supply) requiring climate-informed water-demand projections.

Continental-Scale Climate Bias-Correction, Projection, & Interactive Dashboard

Interactive dashboard showing bias-corrected climate projections across the continental U.S.

Bias Correction (JVBC) • Ensemble Climate Projections • Time-Series Analytics • Interactive Dashboard • Python & xarray • Water & Climate Risk

Quick Facts

Role	Lead Data Scientist / Climate Analytics Engineer
Domain	Climate Risk, Hydrology, Long-Term Planning
Tech Stack	Python, Joint Variable Bias Correction (JVBC), Multi-Model Climate Ensembles, xarray, NumPy, Pandas, Plotly / Dash / Streamlit
Region & Scale	Continental United States (CONUS), ~10 km grid
Live Demo	Tableau Dashboard: US Regional Climate Analysis & Projections

This project built a scalable pipeline to bias-correct, analyze, and visualize multi-model climate projections across the continental United States. Raw global climate model (GCM) outputs exhibit systematic biases and unrealistic spatial patterns that limit their direct use in water-resources, infrastructure, and climate-risk planning.

Using a Joint Variable Bias Correction (JVBC) algorithm, I transformed coarse, biased climate projections into high-resolution, spatially coherent temperature and precipitation fields. The resulting products feed an interactive dashboard that allows planners to explore how key climate variables evolve under different emissions scenarios, time horizons, and user-selected regions.

Background & Problem Statement

Global climate models are powerful tools for simulating future climate, but their native output is often too coarse and biased for regional decision-making. Temperature and precipitation fields can exhibit mean biases, distorted variability, and unrealistic spatial correlation structures when compared against observations.

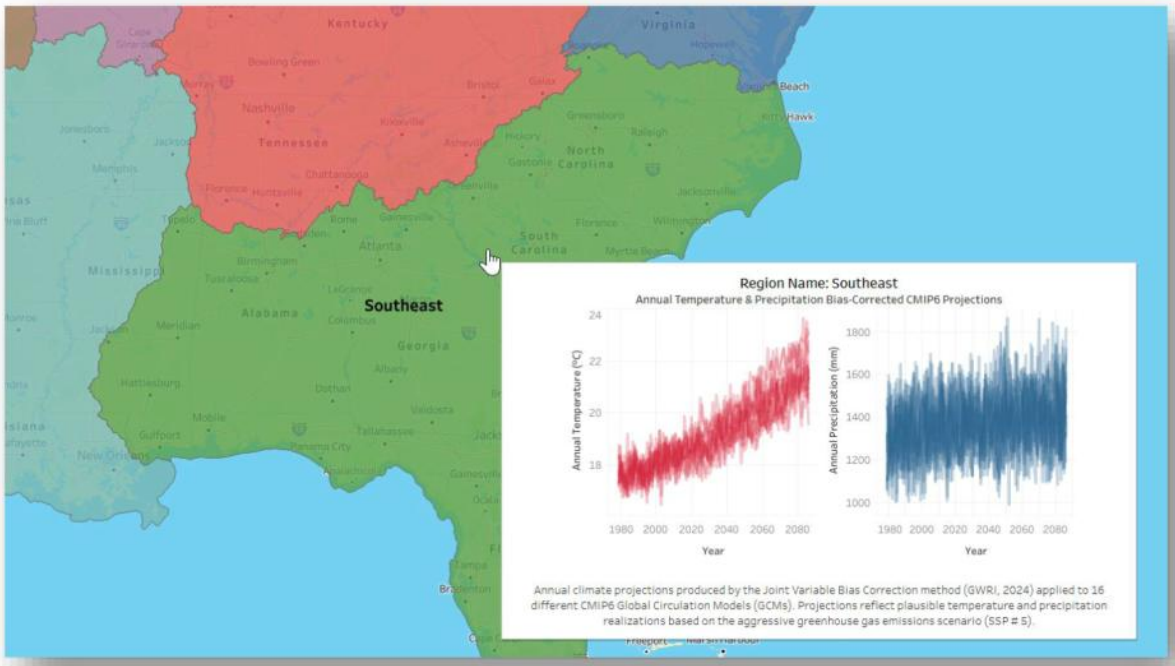
Water and infrastructure planners require climate projections that are both physically plausible and statistically consistent across large regions so they can design robust long-lived assets (reservoirs, conveyance systems, treatment plants, etc.). Ad hoc bias-correction at a few stations is not sufficient for continental-scale assessments.

Problem Statement: How can we convert raw multi-model climate projections into spatially coherent, bias-corrected datasets and a user-friendly dashboard that supports continental-scale climate risk and water-resources planning?

Bias-Correction & Modeling Framework

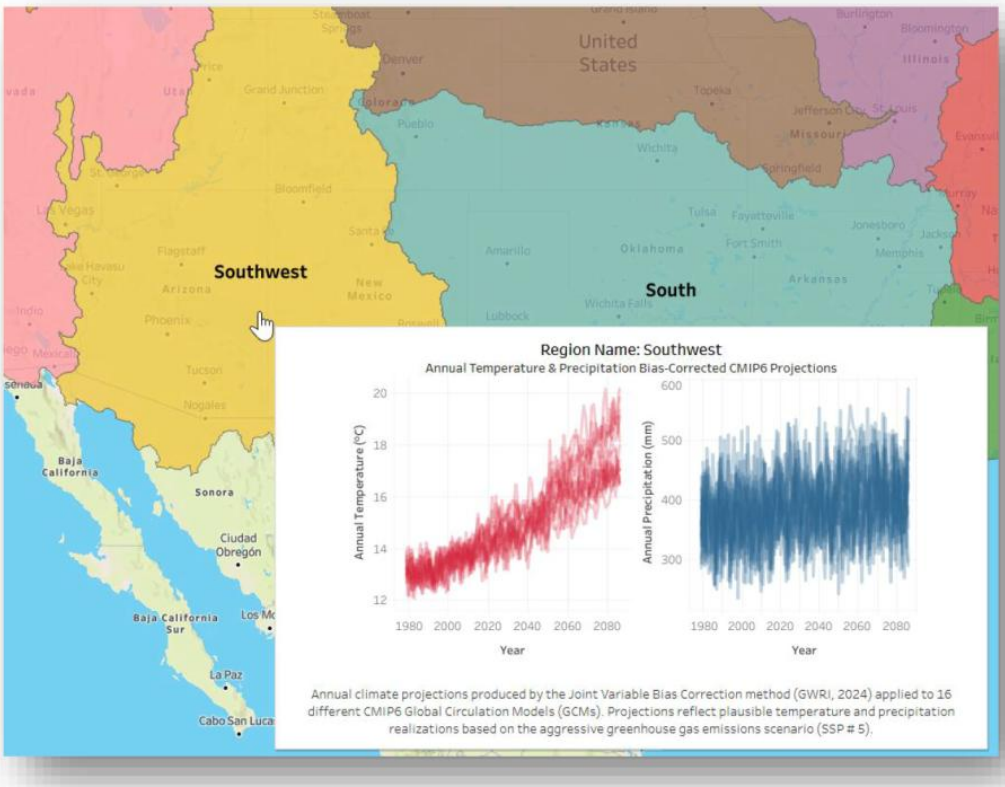
I designed a modular processing pipeline that ingests historical observations and GCM simulations, applies JVBC-based bias correction, and derives climate indicators suitable for planning and communication.

- Data Assembly: Collected multi-model historical and future climate projections (temperature and precipitation) along with gridded observational or reanalysis datasets used as the "truth" reference. All data were harmonized on a common grid and calendar using xarray and dask -friendly workflows.
- Joint Variable Bias Correction (JVBC): Implemented a JVBC algorithm that simultaneously corrects temperature and precipitation, preserving their joint distribution and spatial correlation structure while removing model-specific biases in means, variance, and extremes.
- Temporal & Spatial Scaling: Downscaled coarse GCM fields (~100 km) to ~10 km grid cells and to daily-to-monthly time steps, enabling both local and regional statistics to be computed consistently across CONUS.
- Derived Climate Metrics: Computed seasonal and annual precipitation, mean temperature, hot and cold extremes, and drought-relevant metrics (e.g., precipitation minus potential evapotranspiration).
- Validation & Skill Assessment: Evaluated bias-corrected fields against withheld observations and reanalysis, showing substantial improvement in accuracy and spatial coherence - reducing key error metrics by more than 50% relative to raw model output.



Interactive Tableau Dashboard: Climate Assessment for the Southeast U.S.

Climate Assessment Southeastern U.S.



Interactive Tableau Dashboard: Climate Assessment for the Southwest U.S.

Climate Assessment Southwestern U.S.

Deployment & Interactive Dashboard

Once the core bias-correction and metric-generation pipeline was stable, I focused on packaging the outputs and building an interactive experience that domain experts could explore without touching code.

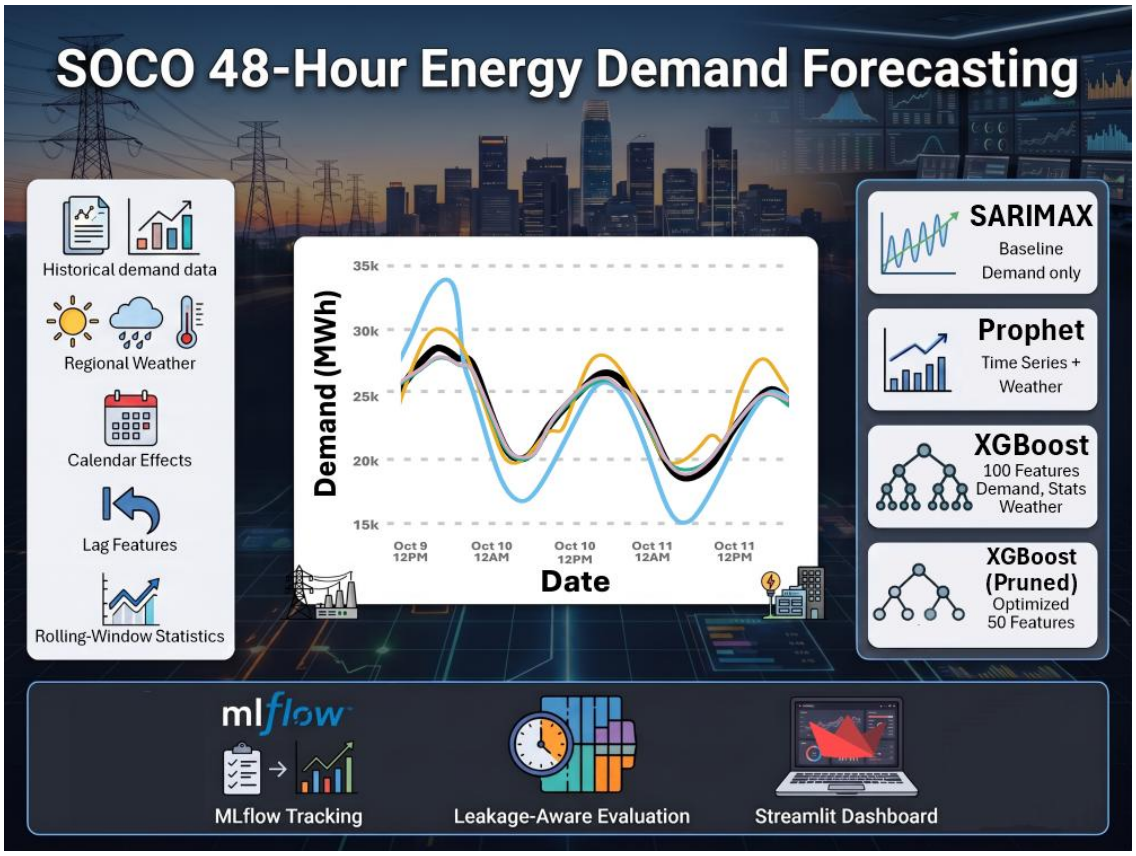
- Pre-Computed Data Products: Exported bias-corrected climate fields and derived indicators as NetCDF and Parquet assets, organized by model, scenario, time horizon, and variable for efficient retrieval.
- API / Data Access Layer: Implemented lightweight access utilities in Python to query time-series or spatial slices for arbitrary locations, watersheds, or administrative regions.
- Interactive Dashboard: Built a web-based dashboard (Plotly Dash / Streamlit) that allows users to: Select emissions scenarios and time slices (e.g., 2030s, 2050s, 2080s). Visualize maps of temperature, precipitation, and drought metrics across CONUS. Extract and plot time-series for specific locations or user-defined regions of interest. Compare raw vs. bias-corrected fields to understand the impact of the correction method.
- Performance & Portability: Optimized data loading and caching so the dashboard remains responsive, and containerized the application for deployment on institutional servers or cloud environments.

Impact & Actionable Insights

The bias-corrected projections and dashboard transform dense climate-model output into a decision-support tool that non-modelers can interrogate directly.

- Improved Reliability: The JVBC framework significantly improves the realism of climate projections, especially for regional precipitation patterns and variability - key drivers of drought and flood risk.
- Scalable to Many Users: Because all heavy computation is performed up front, planners, engineers, and analysts can run "what-if" scenario explorations in the dashboard without waiting on long-running model jobs.
- Supports Long-Term Planning: Utilities and agencies can use the tool to stress test infrastructure and water-management strategies under multiple climate futures, improving the robustness of long-lived investments.
- Reusable Architecture: The same bias-correction and dashboard architecture can be adapted to other regions or sectors (e.g., hydropower, urban heat, wildfire risk), accelerating future climate-analytics projects.

SOCO 48-Hour Energy Demand Forecasting with Time-Series ML



Technical infographic summarizing the SOCO 48-hour energy demand forecasting project

Energy Demand Forecasting • 48-Hour Forecasting • Time-Series ML • SARIMAX • Prophet • XGBoost • Optuna • MLflow • Plotly • Streamlit • PUDL / EIA-930 • Open-Meteo

Quick Facts

Role	Data Scientist / Machine Learning Engineer
Domain	Electric utility grid operations and balancing-authority load forecasting
Objective	Forecast hourly SOCO electricity demand up to 48 hours ahead using demand history, weather, calendar effects, and engineered lag features
Models Compared	SARIMAX baseline, Prophet, XGBoost Full, and XGBoost Pruned Top-50
Dashboard Architecture	Static exported model artifacts power the Streamlit app; the deployed dashboard does not train models, tune models, connect to MLflow, or compute SHAP at runtime
Repository	github.com/helsharif/soco-energy-demand-forecasting

Live Demo	https://soco-energy-demand-forecasting.streamlit.app

This project forecasts hourly electricity demand for the Southern Company (SOCO) balancing authority over a 48-hour horizon. The revamped workflow emphasizes a fair, leakage-aware comparison across classical time-series baselines and supervised machine-learning models using the same modeling dataset, split logic, and evaluation outputs.

The project now includes a polished Streamlit results dashboard, standardized model-result artifacts, MLflow experiment tracking, Optuna tuning, Plotly visualizations, and a pruned XGBoost variant designed to compare the tradeoff between a full feature set and a simpler top-feature model.

Background & Problem Statement

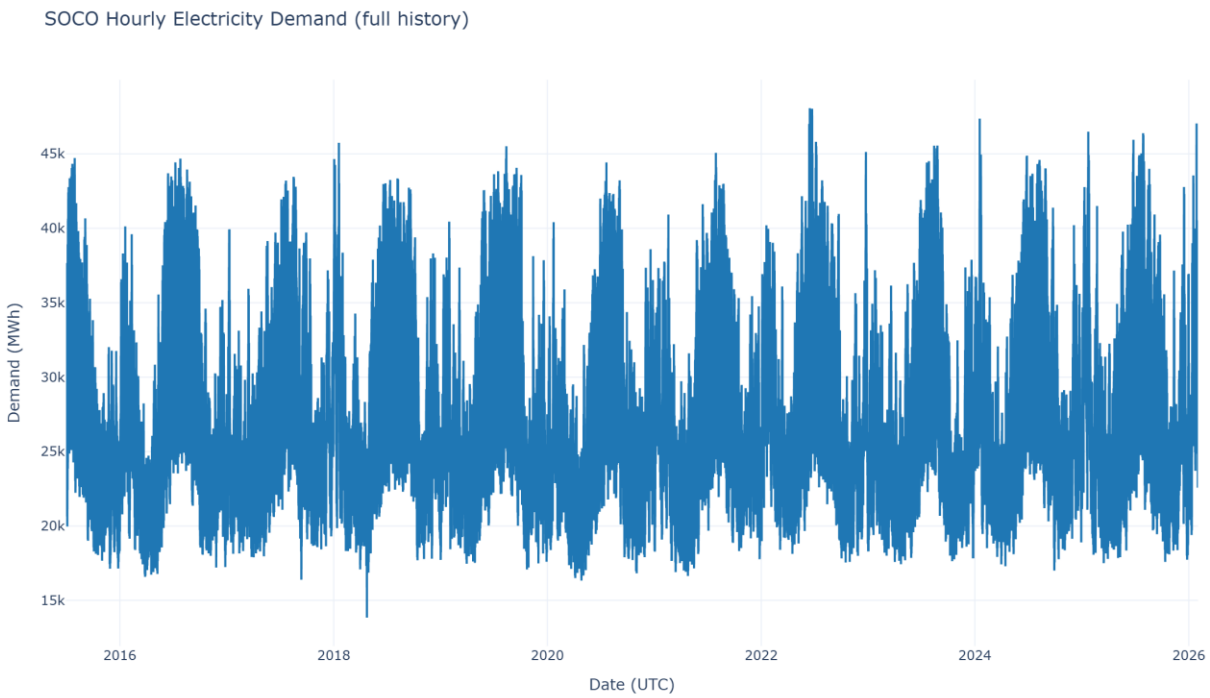
Grid operators need reliable short-term demand forecasts to plan generation commitments, transmission operations, reserve margins, and day-ahead operational decisions. Forecast errors can translate into inefficient dispatch, over-committed reserves, or rapid corrective action when load is higher than expected.

Problem Statement: Given historical hourly SOCO electricity demand and recorded regional weather, can a reproducible forecasting workflow predict hourly demand for the next 48 hours while fairly comparing statistical and machine-learning approaches under leakage-aware evaluation rules?

Dataset & Data Sources

The target variable is demand_imputed_pudl_mwh , an hourly demand series for the SOCO balancing authority derived from PUDL/EIA-930 grid operations data. Weather features are integrated from regional Open-Meteo historical weather records, then transformed into modeling inputs that reflect both daily load shape and temperature-driven heating and cooling effects.

The final modeling dataset supports apples-to-apples model comparison by using consistent timestamps, aligned predictors, and time-aware train, validation, and test splits.



Full SOCO hourly electricity demand history from 2015 to 2026

Feature Engineering

Feature engineering converts raw operations and weather data into horizon-safe predictors for hourly forecasting. The goal is to capture recurring load structure without allowing future demand information to leak into model training or evaluation.

- Lag features: recent demand history such as hourly, daily, and weekly lags that capture autocorrelation.
- Rolling-window statistics: shifted rolling means and variability metrics that summarize recent load behavior.
- Calendar encoding: hour-of-day, day-of-week, weekend, seasonal, and holiday-style effects.
- Weather integration: regional temperature, dew point, humidity, pressure, wind, precipitation, and radiation signals.
- Heating/cooling features: degree-hour style variables that represent nonlinear temperature-demand response.
- Pruned feature set: a simplified XGBoost Top-50 variant for comparing full-feature performance against a leaner model.

Modeling Approach

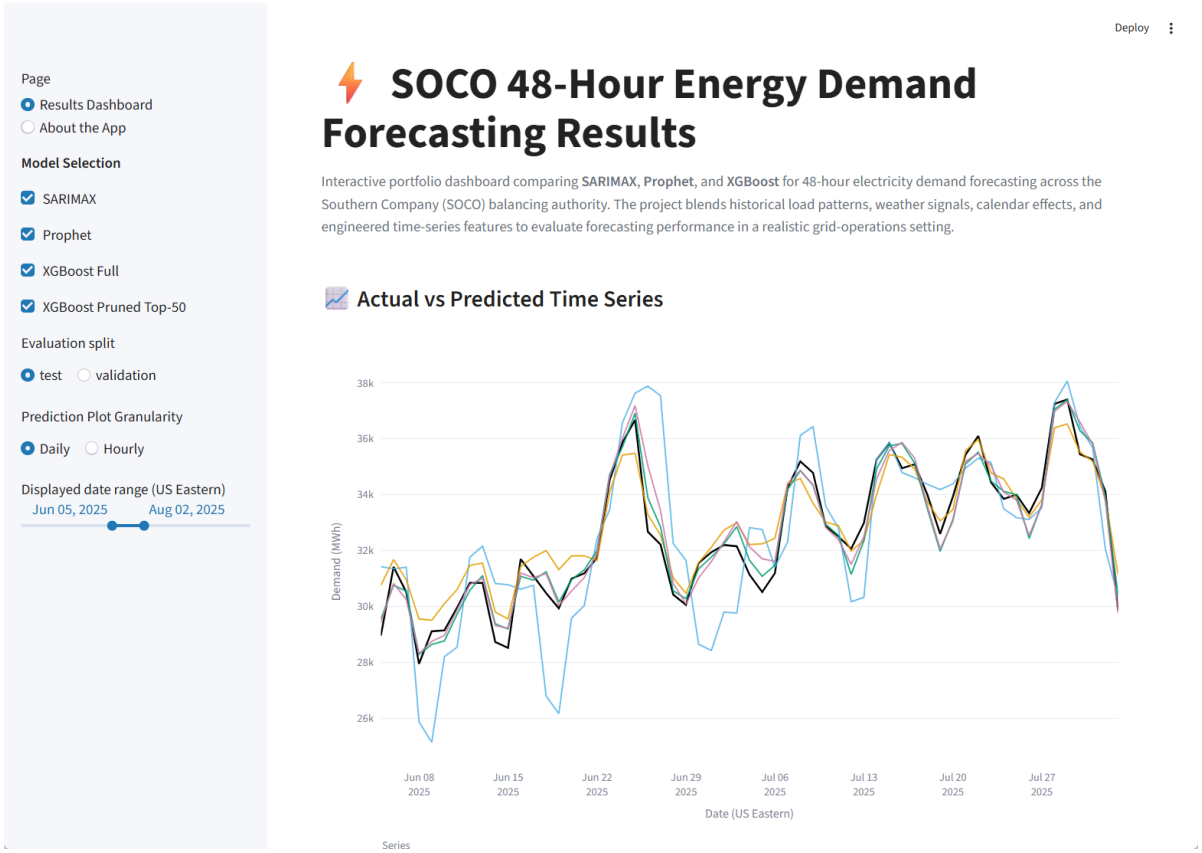
The revamped project compares four model configurations against the same target and evaluation split. SARIMAX acts as the demand-only statistical baseline, Prophet provides a flexible decomposable time-series model, and XGBoost models treat forecasting as a supervised regression problem with engineered demand, weather, and calendar features.

- SARIMAX: baseline statistical time-series model using historical demand structure.
- Prophet: time-series model with trend and seasonality components plus weather-aware regressors.
- XGBoost Full: supervised gradient-boosting model trained with the full engineered feature set.
- XGBoost Pruned Top-50: optimized XGBoost variant using a reduced set of the most useful features.
- Optuna tuning: model hyperparameters are optimized using reproducible search workflows.
- MLflow tracking: runs, parameters, metrics, predictions, and artifacts are logged for comparison and auditability.

Interactive Results Dashboard

The Streamlit app was redesigned as an interactive results dashboard. It starts with the actual versus predicted time-series plot, supports model-selection controls, allows the user to switch between test and validation splits, and recalculates error metrics for the displayed date range.

- Actual vs. predicted: daily or hourly forecast traces compare SARIMAX, Prophet, XGBoost Full, and XGBoost Pruned Top-50 against observed demand.
- Date-range filtering: the visible forecast window can be adjusted interactively from the dashboard sidebar.
- Metric cards: RMSE, MAE, and MAPE are recalculated for the selected time range and split.
- Deployment-friendly data: the deployed app reads static dashboard-ready files rather than training or tuning at runtime.



Streamlit dashboard showing actual versus predicted SOCO energy demand with model controls

Forecast Evaluation & Model Comparison

Evaluation focuses on leakage-aware comparison across the same modeling data and consistent metrics. The dashboard stores standardized per-model result files so that full-test metrics, validation metrics, daily predictions, hourly predictions, horizon errors, feature importance, and SHAP summaries can be refreshed after new MLflow runs.

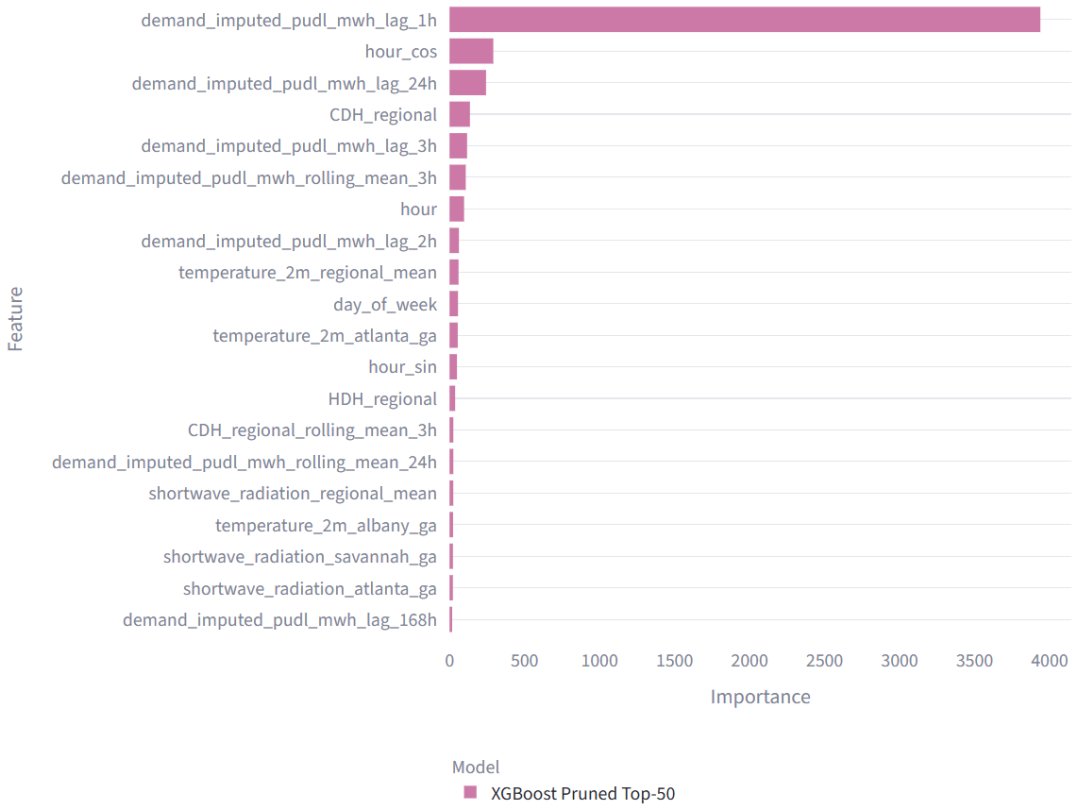
Model Comparison Framework

Model	Primary Role	Input Signal	Dashboard Metrics
SARIMAX	Statistical baseline	Demand history	RMSE, MAE, MAPE, horizon errors
Prophet	Time-series + weather baseline	Trend, seasonality, calendar effects, weather regressors	RMSE, MAE, MAPE, horizon errors
XGBoost Full	Full supervised ML model	Demand lags, rolling statistics, weather, calendar, interaction features	RMSE, MAE, MAPE, feature importance, horizon errors
XGBoost Pruned Top-50	Simplified high-signal ML model	Top retained predictors from the full feature set	RMSE, MAE, MAPE, feature importance, SHAP top features, horizon errors

Interpretability: What Drives the Forecast?

The pruned XGBoost model highlights the dominant predictors behind short-term load forecasting. Recent demand history is the strongest signal, while calendar features and regional heating and cooling indicators add operational context for daily load shape and weather-sensitive demand.

- Recent load dominates: the most important feature is the one-hour lag of demand.
- Daily structure matters: 24-hour lags, hour encodings, and rolling windows capture repeating load cycles.
- Weather adds explanatory value: temperature and degree-hour features help represent HVAC-driven demand.
- Pruning supports communication: the Top-50 model is easier to explain while retaining the core demand, weather, and calendar signals.



Top 20 feature importance values for the XGBoost pruned Top-50 energy demand forecasting model

Dashboard Data Architecture

The deployed dashboard uses a static model-results folder designed for fast loading on GitHub Pages and Streamlit Community Cloud. After new experiments are completed, the best SARIMAX, Prophet, XGBoost Full, and XGBoost Pruned runs are exported into standardized result files and committed with the app.

- model_comparison_metrics.csv: one-row-per-model summary metrics for dashboard cards and tables.
- manifest.json: model names and dashboard metadata.
- Per-model metrics.csv: model-specific evaluation metrics.
- predictions.csv and daily_predictions.csv: hourly and dashboard-friendly daily prediction files.
- horizon_errors.csv: forecast-error diagnostics by prediction horizon.
- feature_importance.csv and shap_top_features.csv: optional interpretability artifacts for tree-based models.

Operational & Portfolio Value

This project demonstrates how to move beyond a notebook-only forecast into a reproducible, reviewable, and stakeholder-friendly ML workflow for energy operations.

- Utility relevance: frames forecasting around day-ahead load planning, reserves, and grid-operations decisions.
- Fair comparison: evaluates multiple forecasting strategies on consistent data and splits.
- Leakage awareness: emphasizes time-aware splitting and horizon-safe feature engineering.
- Model interpretability: uses feature importance and SHAP summaries to explain why forecasts move.
- Production-minded packaging: separates offline training and tuning from lightweight dashboard deployment.

GitHub Repository & Live Demo

The full implementation and public dashboard are available through the links below.

🔗 View Project Repository on GitHub

🚀 Open Live Demo

The repository includes data preparation, feature engineering, model training and tuning, MLflow tracking outputs, static dashboard-ready model-result artifacts, Plotly figures, and the Streamlit portfolio app.