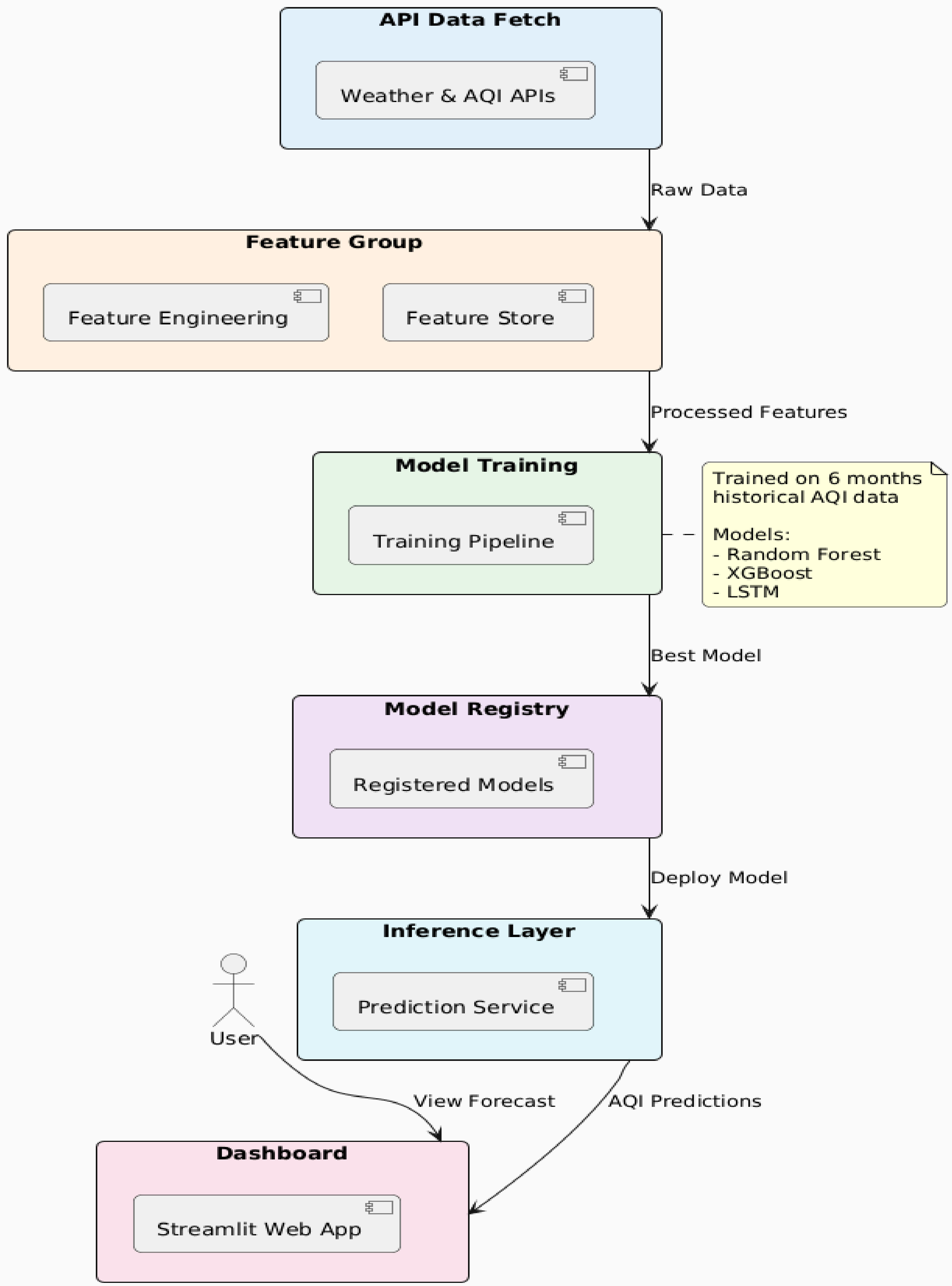
Pearls Karachi AQI Predictor

Presenter: Muhammad Hamza Zeeshan
Data Science, 10Pearls Shine Internship Cohort 7



2. Project Overview

This project focuses on building a **self-sustaining machine learning ecosystem** that predicts air quality for Karachi utilizing a **100% serverless stack**.

Objective

The primary goal is to architect an **end-to-end MLOps pipeline** that:

- **Automates Data Ingestion:** Fetches live pollutant and weather data every hour via API.
- **Ensures Model Freshness:** Implements daily retraining to adapt to new atmospheric patterns.
- **Provides Reliable Forecasts:** Delivers a rolling **72-hour AQI prediction** with high accuracy

3. Data Acquisition

For this project, I integrated the **Open-Weather API** as the primary data provider. It was selected because:

- Up to 40 years back (hourly/daily) Historical Data
- High-resolution **Karachi-specific coordinates**
- Ability to provide atmospheric weather and air quality variables in a single, unified response.

Historical Data Backfilling

To quickly build a training set, I created a **backfilling script** (*backfill_data.py*) to programmatically request **historical data** for Karachi. This yielded **~4,500 records** (**~7 months** of hourly data) before live data collection began.

4. Exploratory Data Analysis (EDA)

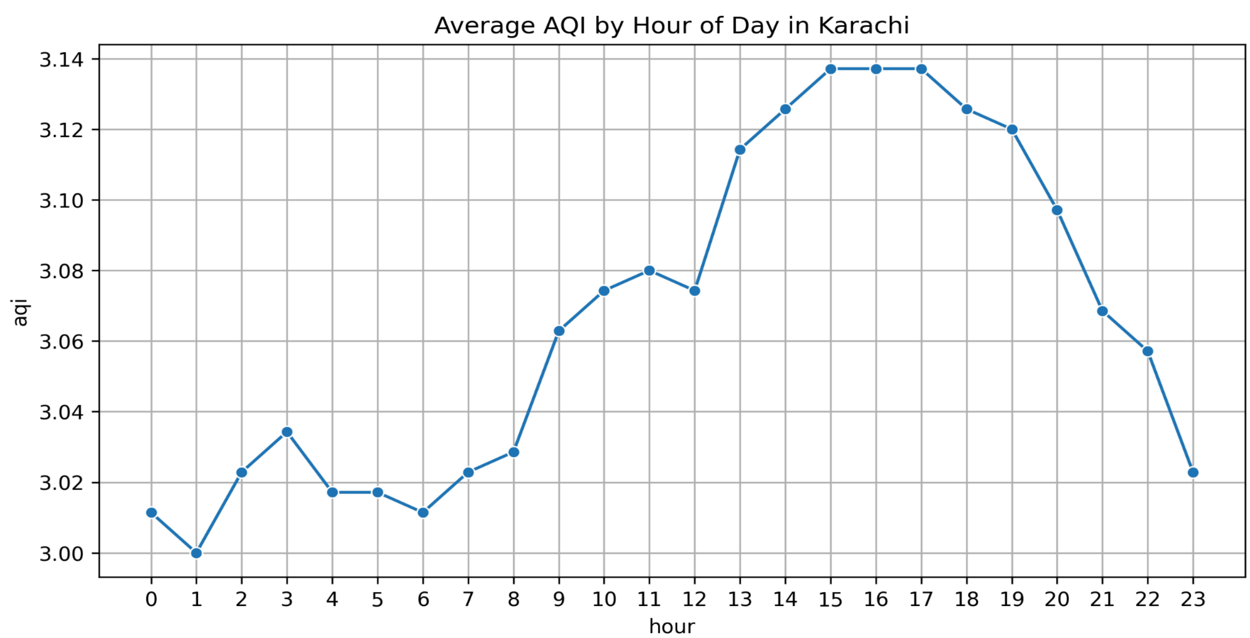
I processed ~4,500 records to create a high-quality baseline dataset for Karachi's air quality forecasting model. This involved transforming raw API data, identifying patterns, and ensuring data integrity.

- **Profiling & Statistical Summary:** Analysis confirmed that AQI levels frequently fluctuate between "Moderate" and "Poor," allowing me to define normal ranges for pollutants like PM2.5 and CO before training.
- **Cleaning & Imputation:** I removed duplicates and handled gaps using imputation techniques.
- **Correlation & Key Findings:** The data revealed a strong direct link between PM2.5 concentrations and the final AQI score.

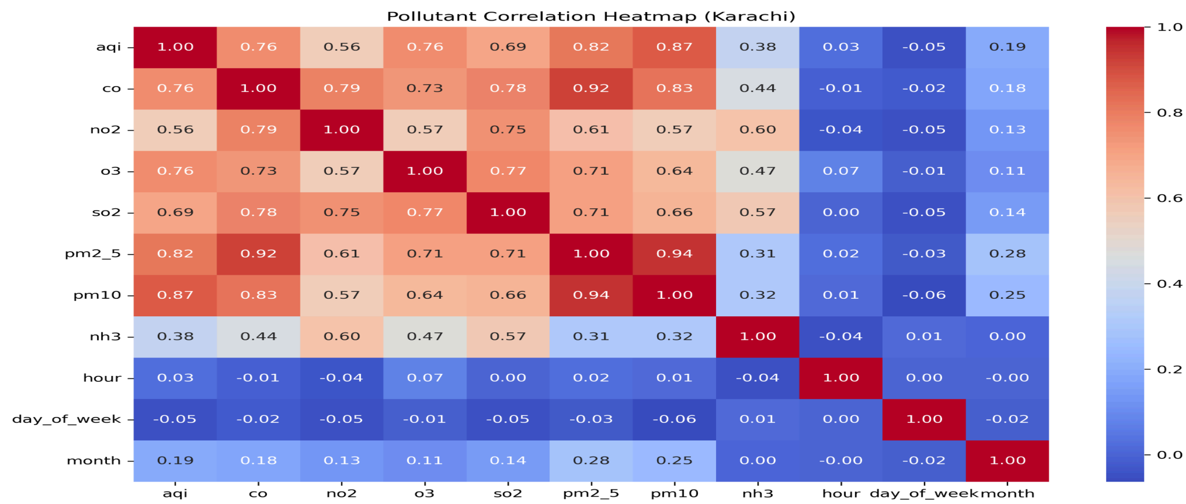
A significant discovery was the **diurnal cycle**, which showed pollution peaks during Karachi’s morning and evening rush hours. These insights confirmed that time-of-day is a mandatory feature for the model to achieve high-accuracy 72-hour predictions.

5. Data Visualization

Time-series line charts showed pollutant levels, especially PM2.5 and CO, spiking bimodally around 9:00 AM and 5:00 PM, confirming the heavy impact of rush-hour vehicular emissions.



Scatter plots and heatmaps revealed that atmospheric conditions influence pollution. A key finding was the inverse correlation between wind speed and AQI: higher wind speeds disperse pollutants, while stagnant, humid air leads to "Hazardous" AQI levels.



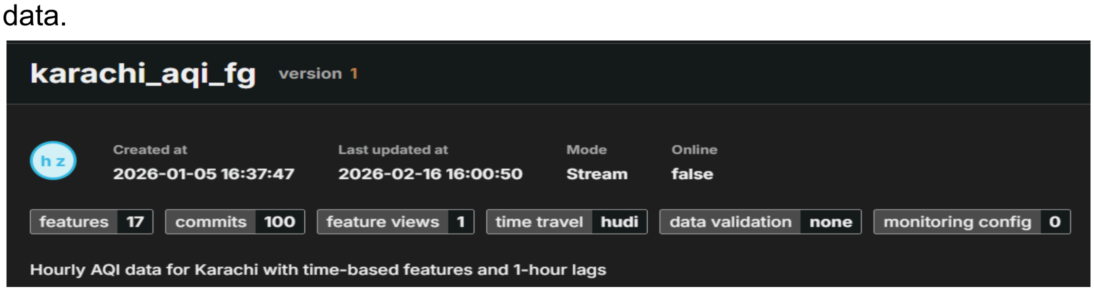
6. Feature Engineering

I engineered a specialized set of **17 features** to capture the complexity of Karachi's atmosphere:

- **Temporal Features:** Extracted **hour, day of the week, and month** to model cyclical traffic and seasonal patterns.
- **Lagged Features:** Created **1-hour lags** for AQI and key pollutants (e.g., aqi_lag_1h), which provide the model with essential "memory" of recent conditions.
- **Dynamic Features:** Calculated the **AQI change rate** to identify how quickly pollution levels are rising or falling.

Feature Store Integration (Hopsworks)

I integrated the **Hopsworks Feature Store** to manage features at scale. It acts as a centralized "data warehouse," with the **Feature Pipeline** automatically inserting hourly updates, guaranteeing both training and inference pipelines consistently access the same, up-to-date data.



7. Model Development

Multiple architectures were benchmarked to find the most effective model:

- **Ridge Regression:** Baseline for linear relationships.
- **Random Forest Regressor:** Selected for superior handling of non-linear patterns.
- **Neural Networks:** DL model to capture long-term atmospheric dependencies.

Training & Hyperparameter Tuning

The models were trained on a historical dataset of **~4,500 records**. To prevent overfitting and ensure the model generalizes well to new data, I implemented:

- **Regularization:** Used in neural architectures to maintain lean, effective weight distributions.
- **Early Stopping:** Automated the training cutoff to stop once the validation error plateaued.
- **Grid Search:** Fine-tuned hyperparameters to minimize the **Mean Absolute Error (MAE)**.

Model Registry (Version Control)

To manage the evolution of the AI, I utilized the **Hopsworks Model Registry** allowing me to:

- **Track Experiments:** Logged over **25 model versions**, comparing their accuracy scores side-by-side.
- **Champion/Challenger Strategy:** Only promoted the best-performing model.
- **Metadata Storage:** Saved training logs and feature importance maps alongside the model file for full transparency.

8. Model Evaluation

8.1 Performance Metrics

I evaluated the model using industry-standard regression metrics to quantify its predictive power:

- **R^2 Score:** This high "Coefficient of Determination" indicates that the model explains over 83% of the variance in Karachi's air quality data.
- **Mean Absolute Error (MAE):** On average, the model's predictions are within a very small margin of the actual recorded AQI values.

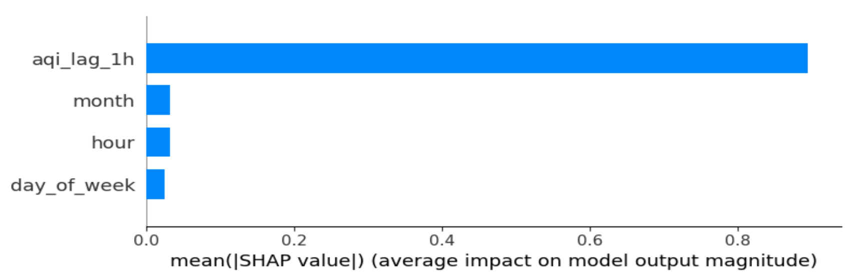
8.2 Prediction Results

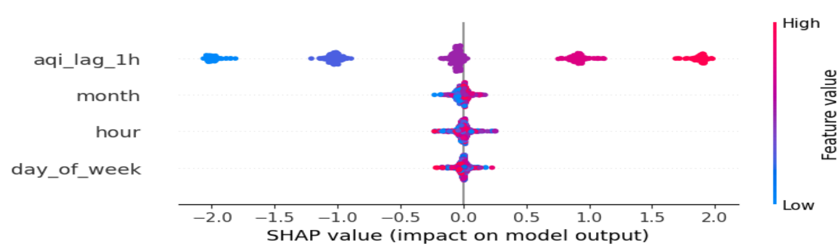
The AI accurately predicts bimodal rush-hour peaks across a 72-hour horizon. Validated on unseen data with high precision, the system is production-ready for delivering reliable early warnings of hazardous air quality.

8.3 Feature Importance Analysis

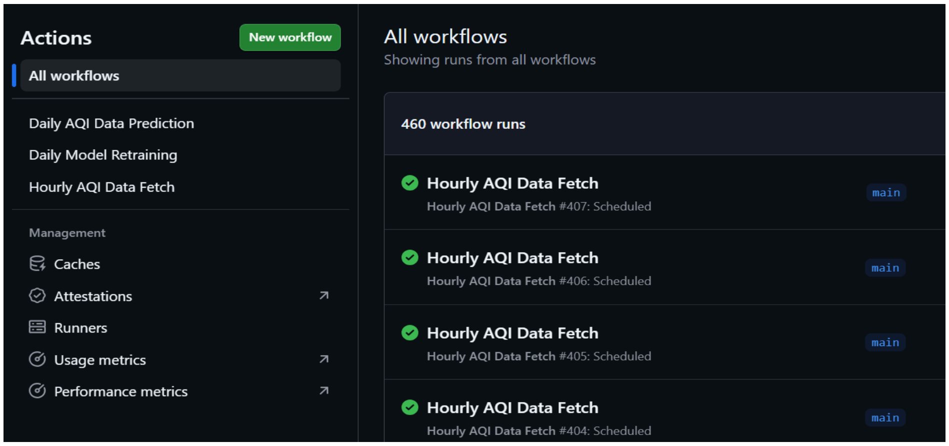
SHAP analysis validated that the model identifies the correct drivers for Karachi's air quality:

- **Primary Drivers: aqi_lag_1h** and **PM2.5** concentrations are the most influential factors.
- **Secondary Signals: Temporal features (hour)** confirm the impact of daily traffic cycles on pollution.





9. Automation & Pipelines



Data Fetch

The hourly **Feature Pipeline** pulls raw API data and processes 17 engineered features directly into the Hopsworks Feature Store.

- **Feature Ingestion:** Updates the model's "memory" with the latest air quality trends.
- **Automation:** Triggered every 60 minutes to ensure data freshness for Karachi.

Model Training

The **Training Pipeline** automates model selection and registration.

- **Chronological Validation:** Utilizes a professional 80/20 time-series split.
- **Benchmarking:** Trains three distinct architectures (**Ridge**, **Random Forest**, and **Neural Network**) with aggressive regularization (high alpha, shallow depth, 40% dropout) to avoid over-fitting on noise.
- **Model Registry:** Automatically selects the "Winner" based on the lowest MAE, exports performance metrics to model_info.json, and registers the model to the Hopsworks Registry.

Inference

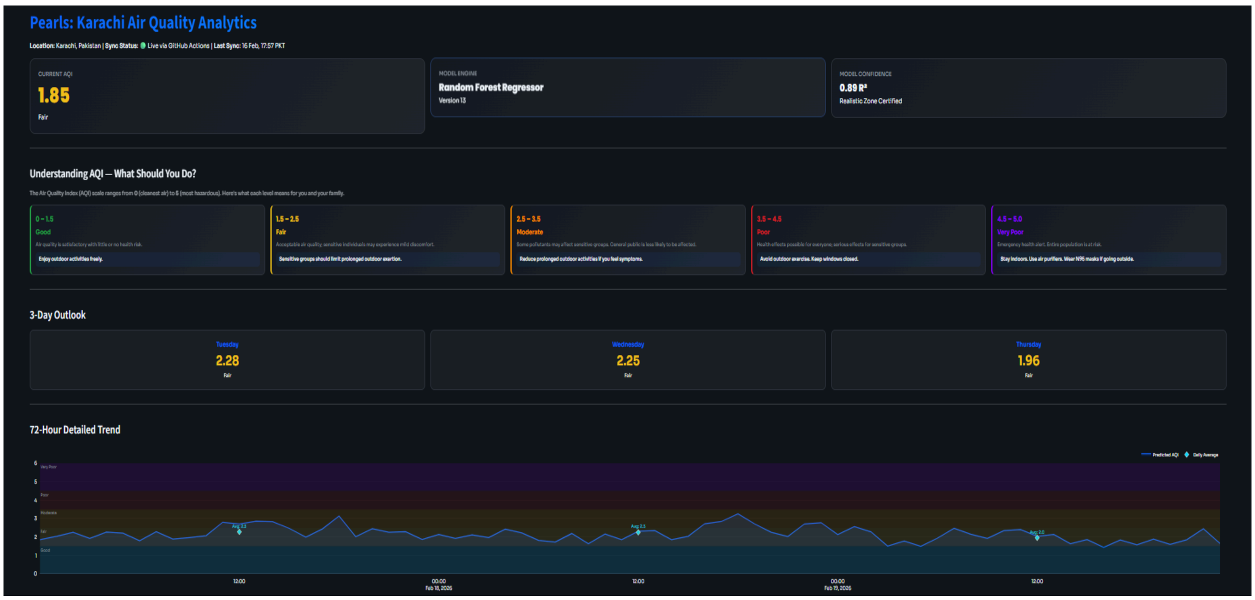
The **Inference Pipeline** converts stored features into actionable 72-hour forecasts. It retrieves the best model from the registry to compute future AQI levels.

- **Batch Forecasting:** Generates a rolling 3-day outlook in a single automated run.
- **Data Delivery:** Pushes results as a CSV back to the repository to act as a live data feed for the Streamlit dashboard.

10. Deployment: Interactive Dashboard

I developed a high-performance Streamlit application to bridge live environmental data with actionable predictive insights for Karachi's citizens.

- **Sober UI Design:** Data readability with a 1–5 color-coded AQI status indicator
- **Performance Optimization:** Implemented st.cache_data to ensure near-instant load
- **Alerts:** Translates raw AQI scores into red-zone warnings and specific advice.



11. Technology Stack

- **Core Logic:** Python 3.11, Pandas, and NumPy for all pipeline data processing.
- **Machine Learning:** Scikit-Learn (Random Forest, Ridge) for modeling and SHAP for interpretability.
- **MLOps Infrastructure:** Hopsworks as the central Feature Store and Model Registry hub.
- **Orchestration:** GitHub Actions for automated hourly/daily YAML-based workflows.
- **Frontend:** Streamlit for the UI and Open-Weather API for real-time meteorological streams.

12. Challenges & Solutions

I resolved several technical bottlenecks to ensure the stability and accuracy of the end-to-end system:

- **Missing Libraries:** GitHub Runners failed because they couldn't find Scikit-Learn.
  Solution: Updated the .yml configuration to ensure all necessary tools are installed automatically before the script runs.
- **Data Type Conflicts:** Hopsworks rejected data because Python's numbers were too large (64-bit vs 32-bit).
  Solution: Added a step to the pipeline to force all data into the correct format before sending it to the feature store.
- **Access Denied:** The dashboard could not load the forecast file. **Solution:** Adjusted repository settings to ensure the frontend had the correct permissions to fetch the results.
- **Schedule Delays:** Automated tasks didn't always start on time at first.
  Solution: Monitored the system behavior for several days to ensure the triggers became stable and consistent.
- **Static Predictions:** The model was "memorizing" data instead of predicting, causing it to output the same numbers.
  Solution: Fixed the overfitting by adding constraints to the model's complexity, forcing it to learn actual patterns.

13. Conclusion: Achievements & Key Learnings

My internship at **10Pearls** was a transformative experience, applying MLOps to a production-grade project within a professional, distributed engineering workflow.

- **Professional Mentorship:** Received expert guidance from architecture design to final deployment.
- **Technical Networking:** Collaborated with a cohort of like-minded computer science students.
- **Serverless Mastery:** Built a scalable system using **Serverless Architecture** without managing physical infrastructure.
- **MLOps Proficiency:** Implemented a **Central Feature Store** and **Model Registry** for professional data/model versioning.
- **Skill Integration:** Synthesized **Python, Data Science, and DevOps** skills into a cohesive solution.