Imperial College London
Department of Earth Science and Engineering
MSc in Geo-energy with Machine Learning and Data Science

Independent Research Project
Final Report

Developing a Fine-Tuned Time Series Model for Automated Test Documentation Generation and Anomaly Detection

Ashwin Vel
Email: ashwin.vel24@imperial.ac.uk
GitHub username: `irp-av1824`
Repository: `https://github.com/ese-ada-lovelace-2024/irp-av1824`

Supervisors:
Andy Nelson
Prof. Martin Blunt

August 2025

Contents

1 AI Acknowledgment Statement

The following generative AI tool was utilized in preparing this report:

- **Tool Name and Version**: ChatGPT-o3
- **Publisher or Provider**: OpenAI
- **URL of the Service**: `https://chat.openai.com`
- **Usage Description**: I've used ChatGPT to generate ideas on how to apply anomaly detection in time series data and to modify user interface screenshots to include additional functionality boxes clearly aligned with existing UI design patterns.
- **Original Work Confirmation**: I confirm that all the submitted work in this report is my own, despite the assistance received from the generative AI tool stated above.

2 Abstract

Manual documentation of well completion tools testing is often time-consuming and prone to errors, which could lead to tool failure. My project aims to automate anomaly detection and quality reporting for well-completion tools by integrating AVEVA DataHub streams into a streamlined web application. The methodology involves developing an Isolation Forest-based unsupervised machine learning model trained on unlabeled sensor data. A Python API will serve anomaly results to TAQA web interface, enabling one-click PDF report generation. If feasible, an LSTM Auto-Encoder will enhance detection accuracy. Expected outcomes include quicker anomaly identification through reduced manual analysis effort and standardized compliance reports demonstrating how advanced anomaly detection can improve operations in the energy sector.

3 Introduction

3.1 Background and Problem Description

Currently, the test data from well completion tools are being stored in AVEVA Datahub such as battery voltage. There isn't any way for engineers who do the testing to automatically validate if the results they've obtained from these test deviates from normal values, which could either signal potential tool failure or a mistake in the testing process.

This process is largely manual which is by plotting or downloading the CSV files. Most of the time, this step is skipped altogether as testers heavily rely on their engineering judgement, increasing the chance of human oversight. This leads to delayed responses and potentially operational risk.

The main goal of my project is to build a web-based system that can detect an anomaly automatically and generate a PDF compliance report. My main challenge in this project is that the data is unlabelled, so there are no ground truth telling if the value is an anomaly. This research explores if machine learning can be used to reliably detect anomalies in unlabelled time-series data and have these findings automatically populated in a compliance document.

3.2 Review of Existing Work

3.2.1 Challenges in Anomaly Detection

The main challenges in anomaly detection for time series data is that anomalies are rare. Because of this, it's hard to evaluate how good a model is, since there's usually no proper ground truth. The best we can do is get domain experts to label some of the data, but even that is subjective. In unsupervised anomaly approaches where anomalies show up as deviations from predictions, it is found that most thresholding methods (mean error and standard deviations) don't work well in practice [1]. This shows there's still a lot of room for improvement in how we set thresholds in anomaly detection.

3.2.2 Model Selection

Isolation Forest (IF) and Support Vector Machines (SVM) are among the most commonly used models for anomaly detection in industrial sensor data, particularly when labelled data is scarce. Isolation Forest, in particular, is favored due to its linear time complexity, low memory usage, and simple implementation [2].

However, traditional models like IF may produce high false positive rates in datasets with strong seasonal patterns. In such cases, deep learning-based approaches such as Long Short-Term Memory (LSTM) autoencoders [3] and Deep Support Vector Data Description (Deep SVDD) [4] offer better accuracy but with significantly higher computational demands and complex hyperparameter tuning requirements.

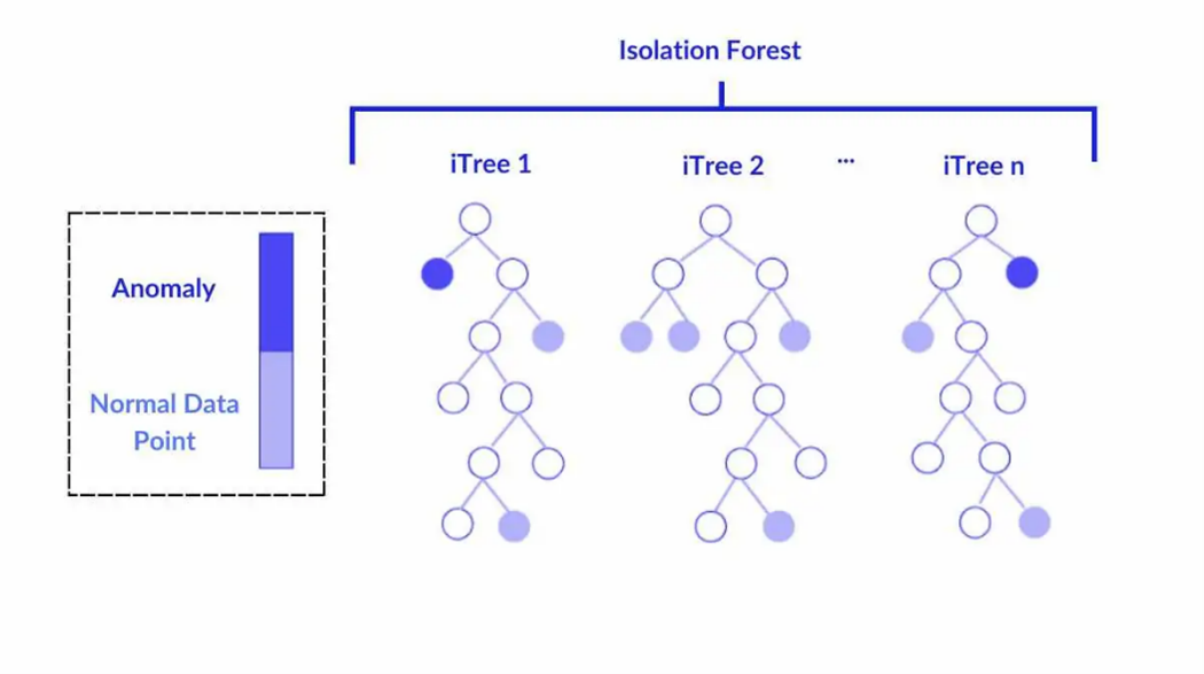

Figure 1: Isolation Forest Architecture

The novelty of this project lies in developing an integrated web-based platform that brings together DataHub ingestion, unsupervised anomaly detection, and one-click PDF report generation within the existing TAQA interface for well completion tool validation.

3.2.3 Oil & Gas Case Studies

IF are introduced in recent oilfield applications where the algorithm is used to predict ESP failure in water injection wells. Results show IF could predict accurately 21 out of 45 ESP failure events when tested on historical data [5]. Febrita et al. developed a deep learning method using LSTM to detect anomaly using real time data [6]. If the delta of Loss MAE between historical and current data exceeds a set threshold, it flags an anomaly.

Other methods include using a stacked spatial-temporal autoencoder (Sst-AE) to enhance spatiotemporal features in industrial multi-sensor data [7]. An adaptive dynamic thresholding method trains the model, and anomalies are identified by comparing enhanced features against cluster centers derived from high-dimensional K-means clustering.

3.3 Objectives

The project will include software development, API integration, and the application of a time-series model for anomaly detection in an industry setting. The key objective:

- Creating a fully operational web-application allowing users to generate test documentation based on minimal user input.

- The application needs to integrate with data storage though external APIs.
- Anomaly detection module for identifying irregularities in test outcomes and comparing test results to historical baseline.

3.4 Significance

Documentation of quality test results is a critical step in ensuring well completion tools comply with industry standards. Manual documentation is often time-consuming and prone to errors. Automating this process will not only save time but also enhance the reliability of compliance checks.

Additionally, leveraging historical test data for anomaly detection can enhance the system's capabilities in detecting faulty equipment and design irregularities, reducing human-prone judgment error.

4 Methodology

4.1 Data Acquisition

The primary data source for this project will be from AVEVA DataHub since all the historical data from the past SIT had been stored there. Access token for AVEVA DataHub will be read from the `appsettings.json` file, so the C# backend will create an authenticated client.

Before downloading the whole dataset from the cloud, I created a landing page in the TAQA webapp (`/Explore`) using AVEVA API keys to scroll through all the SIT data that is available to have a quick glance through what is available. Training on all streams would be time consuming and not part of the project scope. For proof of concept, 8 concrete streams of data will be selected.

The main reason I created this landing page was so that I can confirm if the data exists before modelling and I can identify all the clean, stable streams for training. This will avoid me from training on empty/mislabelled data.

Based on the exploration and the advice of the engineer doing the testing we've decided:

1. Which tools will be selected for training (9 tools were available)
   (a) P8-1, P8-7, P8-11, P8-41, P8-59
2. What streams will be used to train the models
   (a) Battery-Voltage
   (b) Upstream Pressure & Temperature

(c) Downstream Pressure & Temperature

(d) Choke-Position & Target Choke Position

(e) ToolStateNum

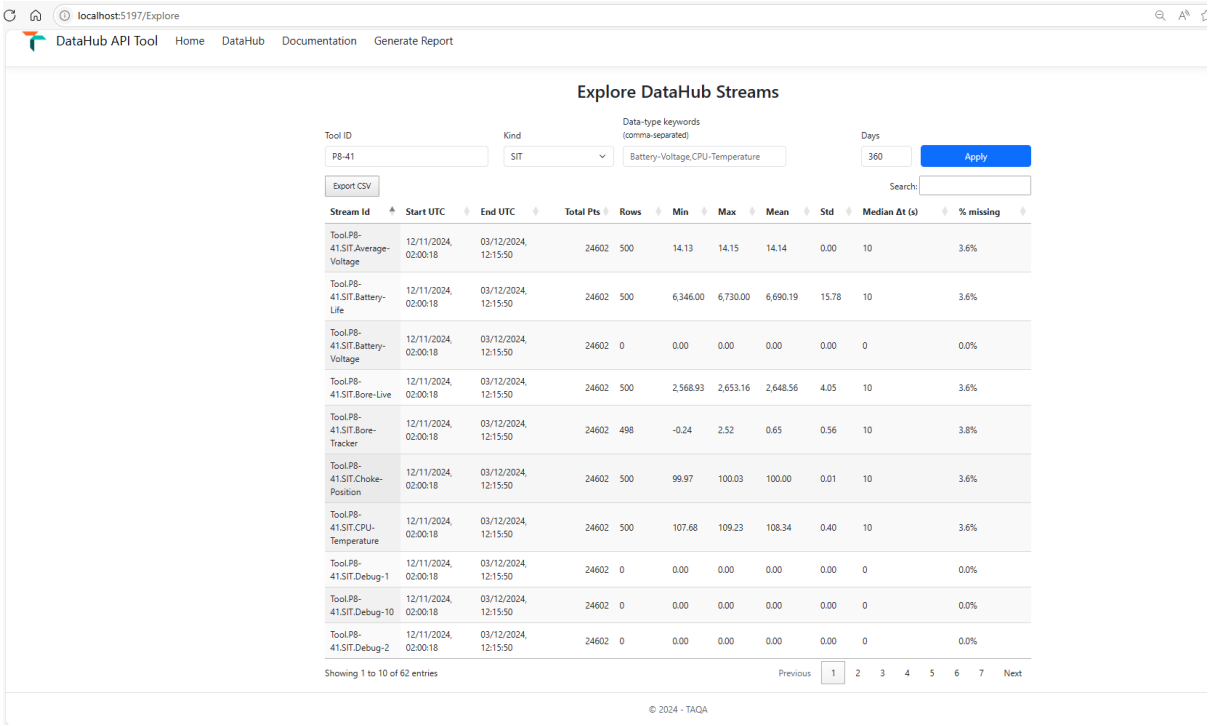

Figure 2: Explore view used to browse SIT streams in AVEVA DataHub and shortlist training data.

One of the main issues I faced when trying to pull the information from the cloud was SDS metadata searches kept failing, showing `Bad Request`. I finally found out that I was violating SDS query grammar as I was mixing the dots and wildcards incorrectly. I solved it by enumerating the streams with coarse SDS filters (e.g., `Id:Tool.*`) and then parsed the tool via regex which avoided the 400 errors.

4.2 Exploratory Data Analysis

After shortlisting the streams I will be using, the data was downloaded from the current landing page of the TAQA webapp in CSV format using `Export CSV` button. All the tool data were concatenated into one large dataframe and saved in parquet format for efficient processing. Here is what is learned from the EDA that led to my preprocessing choice:

1. **Temporal gap analysis:** 98% of data had 1 s data frequency except for P8-41 which had 1.3% of data > 10 s.

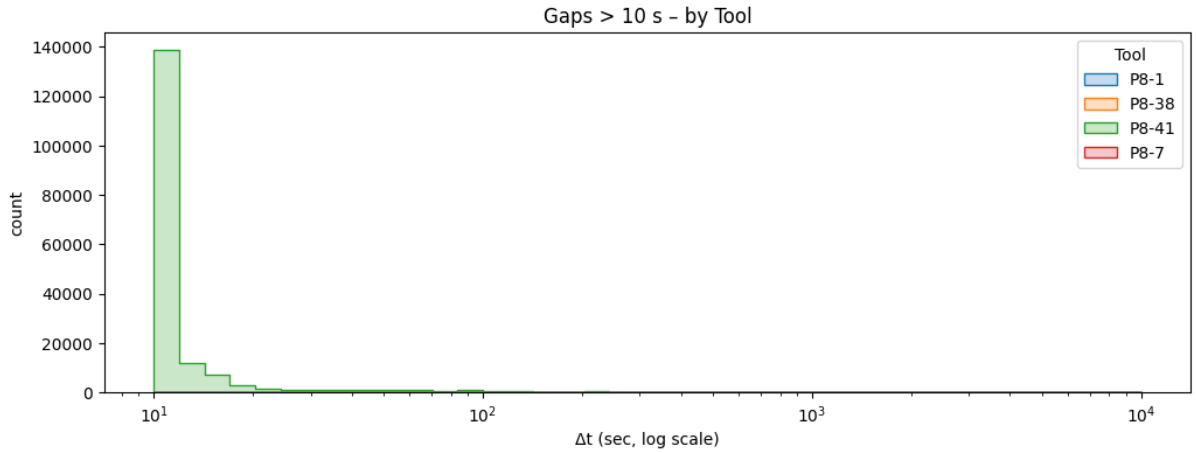

Figure 3: Temporal gap analysis across tools; P8-41 shows occasional gaps $> 10\,\text{s}$.

2. **Data distribution:** Deciding what hard-limits to run before ML and which scalers are right for the data. Pressure data is very left skewed (skewness > 3.8) even after removing data post hard-limits.

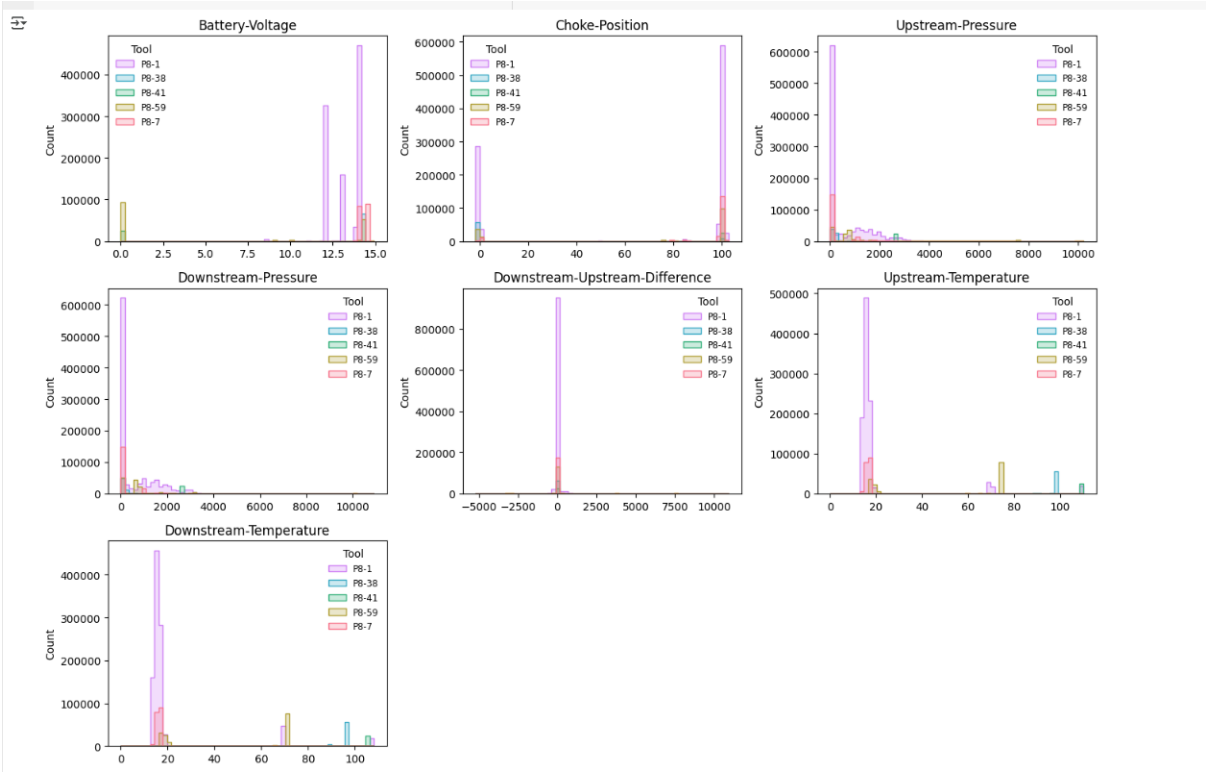

Figure 4: Distributions informing hard-limits and scaler choices (log1p for pressure).

3. **Sensor correlation:** To establish feature importance. Found upstream and downstream temperature are effectively identical while pressure showed positive correlation.

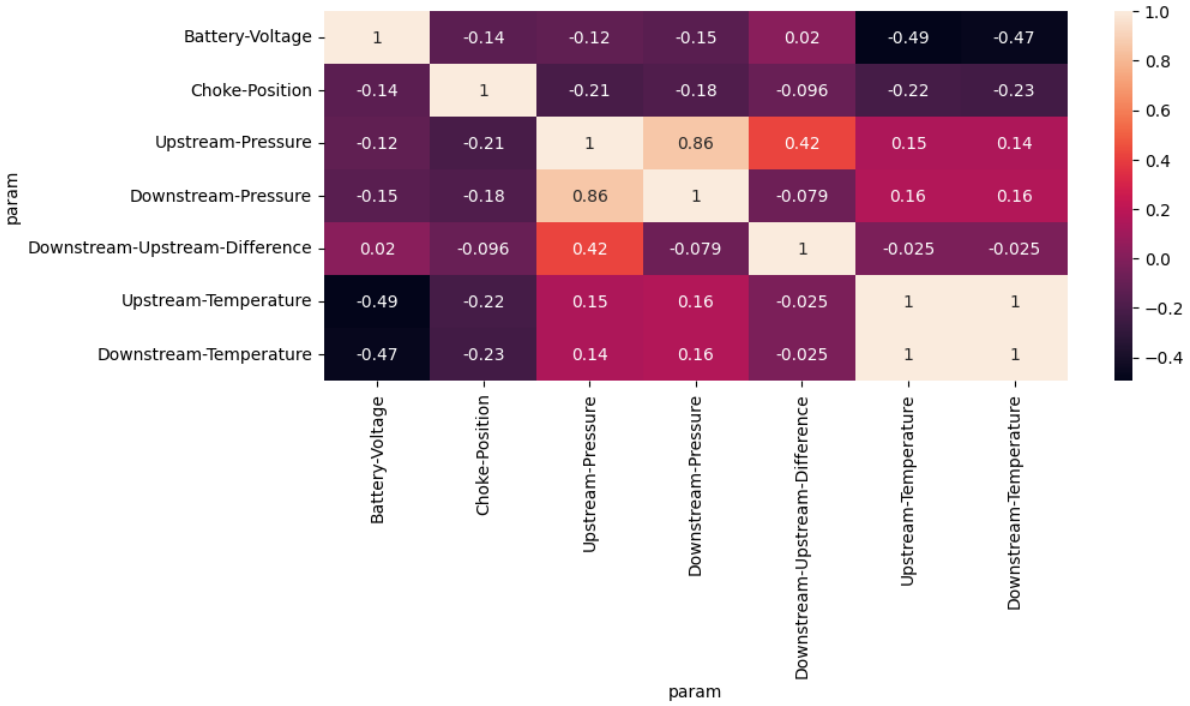

Figure 5: Correlation analysis of key sensors (temperatures nearly identical; pressures positively correlated).

4. **Data volume & coverage assessment:** Quantify row counts and data span. Identified 0.4% data with NaN.

param	min	median	max	std
Battery-Voltage	0.000000	14.139100	14.94505	3.841086
Choke-Position	-1.891891	100.513500	102.86490	46.634118
Downstream-Pressure	0.000000	22.681820	10932.98000	1012.014208
Downstream-Temperature	0.000000	16.280880	108.21590	26.618914
Downstream-Upstream-Difference	-5209.666000	-1.644318	10925.18000	569.151351
Upstream-Pressure	0.000000	30.935460	10214.19000	1097.800281
Upstream-Temperature	0.000000	16.349700	110.10000	27.357168

Figure 6: Coverage and volume overview; NaN proportion $\approx 0.4\%$.

4.3 Bottom-line preprocessing:

1. Timestamp standardization (UTC-align) — combine all features by timestamp for robust time-series analysis.

2. 1 s resampling and forward-fill up to 10 s → incorporate P8-41 gaps.

3. Drop constant/NaN rows (∼21%) — constant values to 3 d.p. indicate sensor malfunction.

4. Only one model trained for temperature → effectively the same feature.

5. Use log scaling (`log1p`) of the pressure data; rest use `RobustScaler` → reduced skewness from > 3.8 to < 1.0; RobustScaler is less sensitive to outliers than StandardScaler.

6. Hard-limits (battery < 10 V; pressure < 0 psi or > 7000 psi; temp < 0°C or > 150°C) → removes obvious outliers for cleaner training.

7. Feature engineering → additional physics features (ΔTemperature as divergence signal; `IsOpen` boolean for choke position $> 10\%$).

4.4 Model Development & Training

4.4.1 Model Architecture

For the base case, two different machine learning model architectures were selected. This project developed into an ensemble strategy where we used a multi-model approach to capture the various types of anomalies present within the testing. The main choice of choosing Isolation Forest (IF) and XGBRegressor boiled down to strong performance for unsupervised anomaly detection with high-dimensional sensor data and ease of ONNX export for C# backend inference. These models are lightweight so inference runs in < 10 ms/row. Summary of the 10-model ensemble is shown below:

Table 1: Ensemble Model Architecture

Model	`Input`	**Size**	**What it detects**
1D IF — ΔTemp (open/shut)	`DeltaTemperature`	1.18 MB	When there is a seal leakage, will also be regime aware
2D IF Pressure Pair	`Upstream-Pressure`, `Downstream-Pressure`	1.75 MB	Alerts when pressure gauges diverge
3D IF Choke Position	`Choke-Position, ToolStateNum, Downstream-Temperature`	1.57 MB	Jitters in choke position
7D IF Full Vector	`Battery-Voltage`, `Upstream-Pressure`, `Downstream-Pressure`, `DeltaPressure`, `Upstream-Temperature`, `Downstream-Temperature`, `Choke-Position`	1.76 MB	For never-before-seen operating states; multi-sensor anomalies
6 XGB-Residual models	`For feature y, uses other 6 features as input`	0.20–0.67 MB	One-sensor anomalous vs. others normal
Simple DQ watchdog	`Rule-based`	–	Instantly flags off-scale readings

4.4.2 Model Outputs

1. IF:
   (a) `output_label` (+1 normal, –1 outlier)
   (b) `output_probability` (decision score; $< 0 \rightarrow$ anomaly)
2. XGBRegressor:
   (a) Predicts output
   (b) Compute residual = |obs − pred|
   (c) Alert when residual > cut-off

The cutoffs for the XGBRegressor were calculated using STL+MAD for robust thresholding. For Battery-Voltage, the residual cut-off was derived as 5×MAD, capturing the median swing within the dataset.

4.4.3 Hyperparameter Tuning

For IF models, contamination rate (0.15–0.25) and estimators (200–300) were tuned via grid search to optimize sensitivity while minimizing false positives. For XGBRegressor,

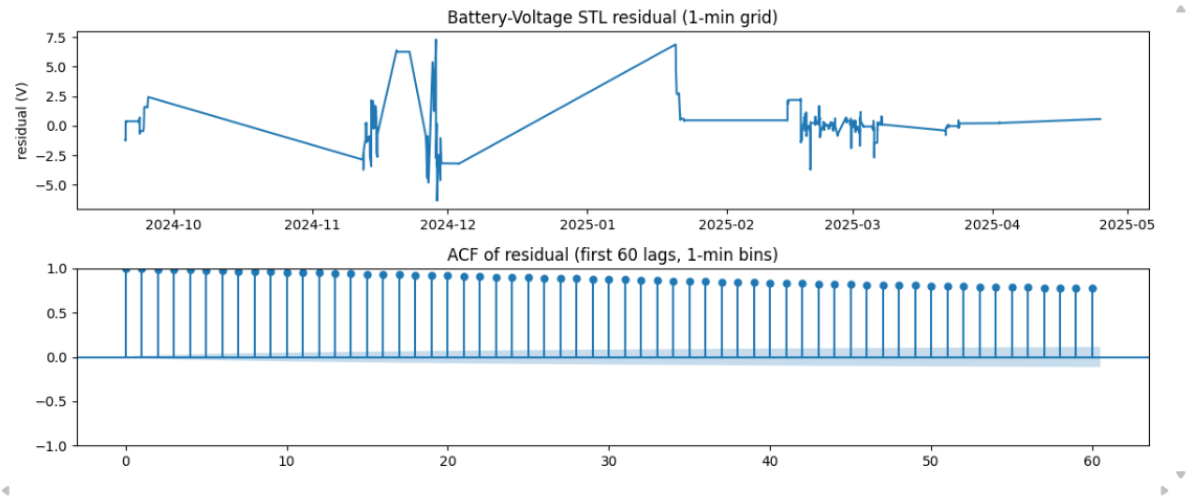

Figure 7: STL+MAD thresholding for residuals (example: Battery-Voltage).

a `RandomizedSearchCV` over key hyperparameters selected the best settings, which were then used in the optimized run.

4.4.4 Model Evaluation

With no native labels, I worked with a domain expert to define what constitutes an anomaly. We then injected synthetic anomalies into a clean, verified slice to obtain ground truth for quantitative assessment. I created tiered evaluation using easy/medium/hard anomaly categories to assess performance across severities.

With the labelled subset (∼500 samples), we evaluated precision, recall, F1, and PR-AUC, emphasizing F1 optimization to reduce false negatives (crucial for downhole valve health).

Additionally, an MLflow-based tracking system monitors production performance with alerts on degradation, making it easier to track which hyperparameters and preprocessing yielded specific improvements.

4.5 Model Deployment

The models are deployed locally and packaged using FastAPI. For future retraining, this remains in the Python environment. Inference uses Microsoft ML.NET ONNX Runtime. Batched processing improves CPU utilization, reducing inference from 10 minutes to 2 minutes (∼80% improvement).

Two data-ingestion options are provided for engineers:

1. CSV upload (drag & drop)
2. Direct pull from DataHub

Once the data is fed, several preprocessing steps run in the .NET backend to prepare for inference:

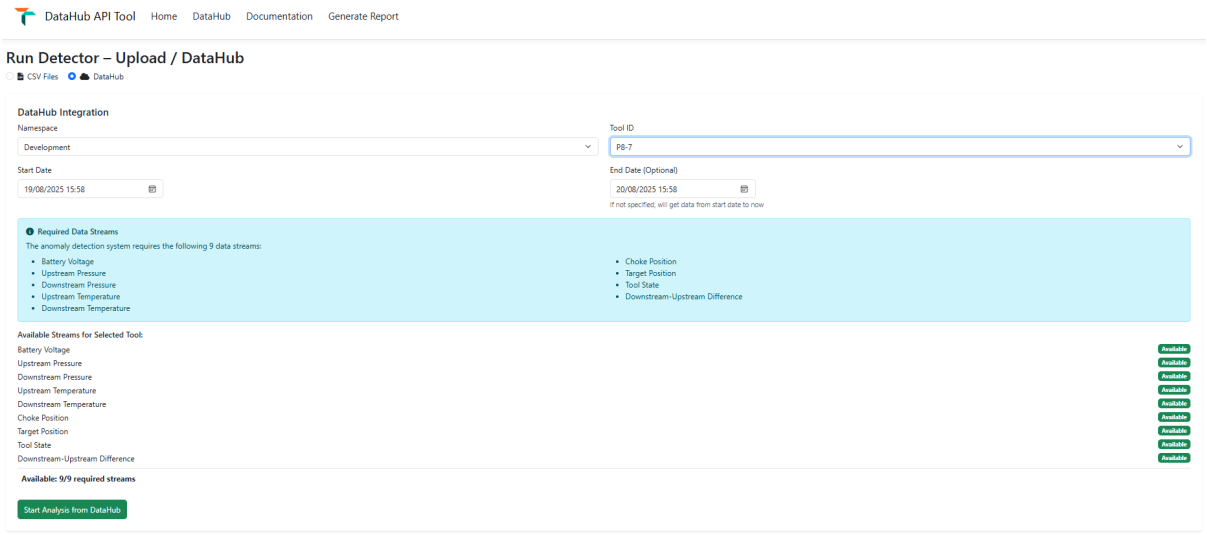

Figure 8: Local API endpoint and data-ingestion UI (CSV upload and DataHub options).

1. Align sensors by timestamp
2. Automatically remove easy outliers using DQ watching (rule alerts)
3. Apply the same feature engineering and scaling as training
4. Batch preprocessing for efficient inference

A key issue was timestamp alignment. Some datasets lacked exact matching stamps; I switched `MultiCsvJoiner.Join()` from strict joins to a gap-tolerant outer join (union of timestamps with 2 s tolerance).

After preprocessing, the models are called via a local HTTP endpoint. An anomaly severity alert maps scores to High/Medium/Low via model-specific thresholds.

4.5.1 Dashboard UI

An interactive dashboard summarizes anomalies via KPI cards and charts. The dashboard is a Razor Page using Bootstrap for responsive layout and Chart.js for visualizations. Key features:

1. Paginated DataTable listing anomaly occurrences
2. Severity mix chart
3. Sensor health strip for data-quality issues
4. Drill-down modal per anomaly for detail

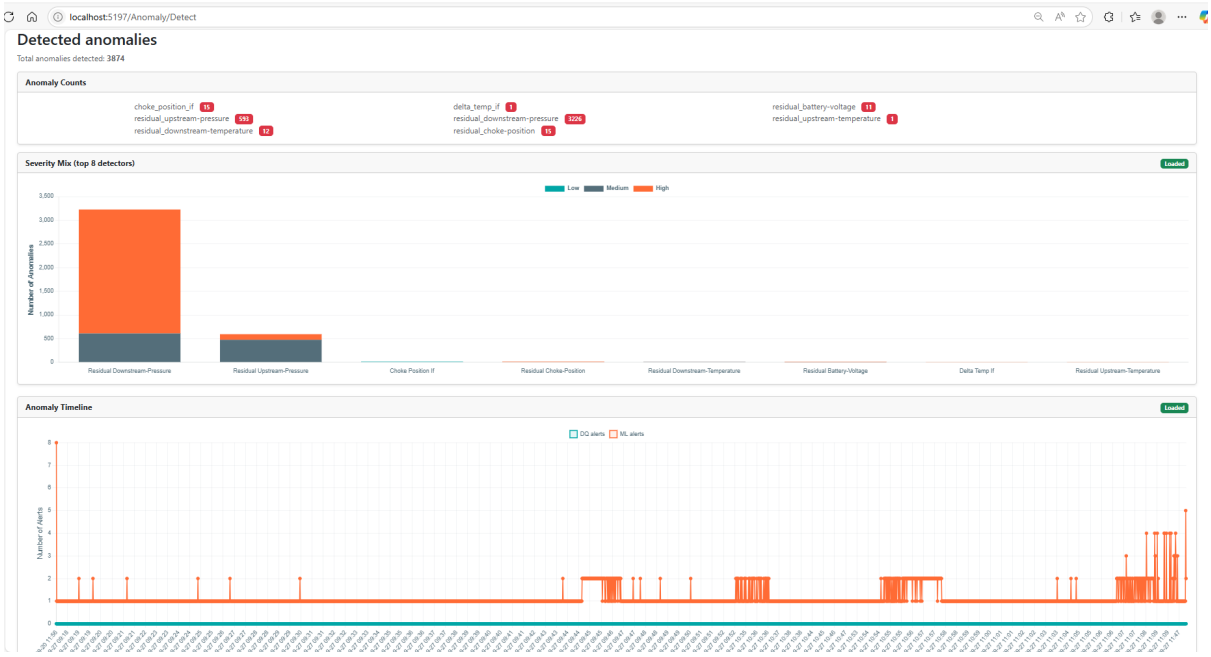

Figure 9: Anomaly dashboard UI with KPIs, charts, and drill-downs.

4.5.2 Automated PDF Generation

PDF generator added via QuestPDF. JavaScript gathers chart images (base64), summary stats, and tables, serializes to a JSON (`reportData`), and posts to `GenerateReport`. On the server, `AnomalyController.cs` receives `AnomalyReportData`, renders the PDF in-memory, and returns it as a browser download.

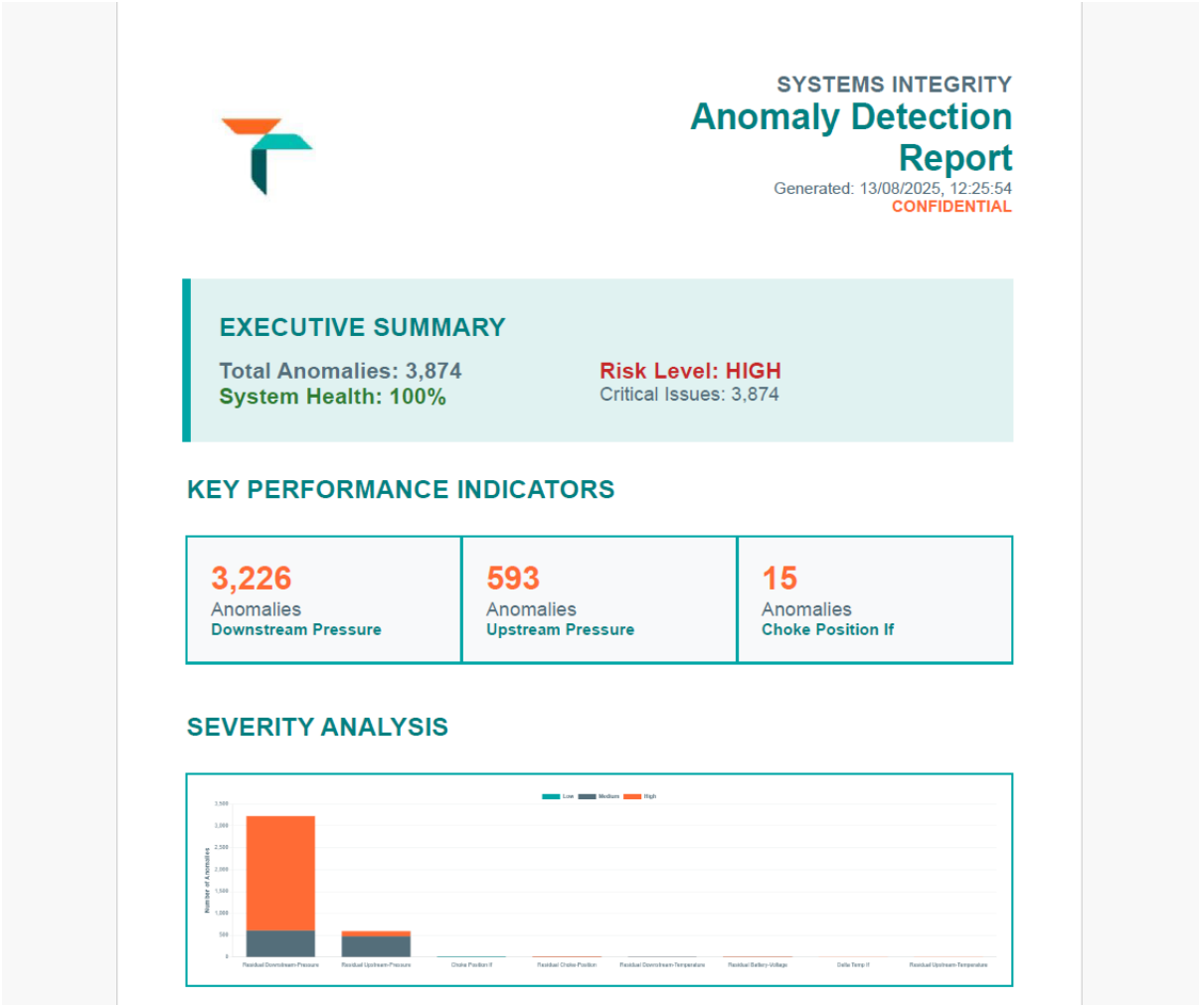

Figure 10: Example of the generated PDF report with charts, tables, and summary.

5 Results

5.1 Model Performance

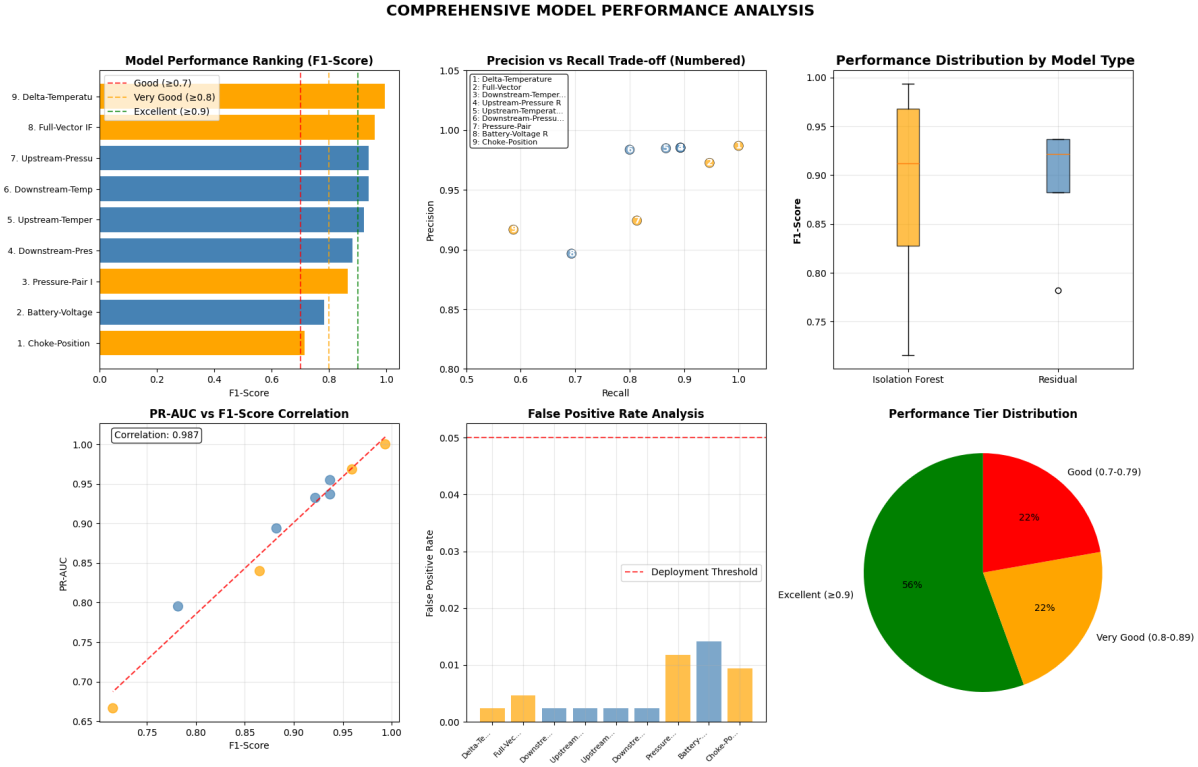

Figure 11: Model Performance Across the Board

Table 2: Ensemble Model Performance

Model Type	**Model Name**	**Target**	**F1 Score**	**Precision**	**Recall**	**PR_AUC**	**FP_Rate**	**Threshold**	**Rank**
Isolation Forest	Delta-Temperature IF	Delta-Temperature	0.9934	0.9868	1.0000	1.0000	0.0024	-0.3299	1
Isolation Forest	Full-Vector IF	Multi-Feature	0.9595	0.9726	0.9467	0.9683	0.0047	-0.0664	2
Residual	Downstream-Temperature Residual	Downstream-Temperature	0.9371	0.9853	0.8933	0.9368	0.0024	1.3127	3
Residual	Upstream-Pressure Residual	Upstream-Pressure	0.9371	0.9853	0.8933	0.9547	0.0024	266.1267	4
Residual	Upstream-Temperature Residual	Upstream-Temperature	0.9220	0.9848	0.8667	0.9322	0.0024	1.5120	5
Residual	Downstream-Pressure Residual	Downstream-Pressure	0.8824	0.9836	0.8000	0.8937	0.0024	299.5785	6
Isolation Forest	Pressure-Pair IF	Pressure-Pair	0.8652	0.9242	0.8133	0.8397	0.0118	-0.0985	7
Residual	Battery-Voltage Residual	Battery-Voltage	0.7820	0.8966	0.6933	0.7951	0.0141	1.2989	8
Isolation Forest	Choke-Position IF	Choke-Position	0.7154	0.9167	0.5867	0.6664	0.0094	-0.0356	9

Eventhough the model was trained on unlabelled data, I tested the model on 500 rows of hand-labelled data to see how the model performed. P-36 tool was chosen to be used as the test data (it was not included in the training). This was also a test of how the model performs over different pressure/temperature regimes. From the model performance ranking chart, we can see that Delta-Temp IF (only using 1 feature) achieved the highest F-1 score (0.9934). This is quite surprising for me as it outperformed complex 9-feature models (though it was feature engineered using 2 features). Normally, we would expect increasing the number of features would improve anomaly detection as it starts to understand context but this just goes to show that well-selected physics based features can be far more effective than high-dimensional approaches in downhole tools.

5.2 Feature Importance

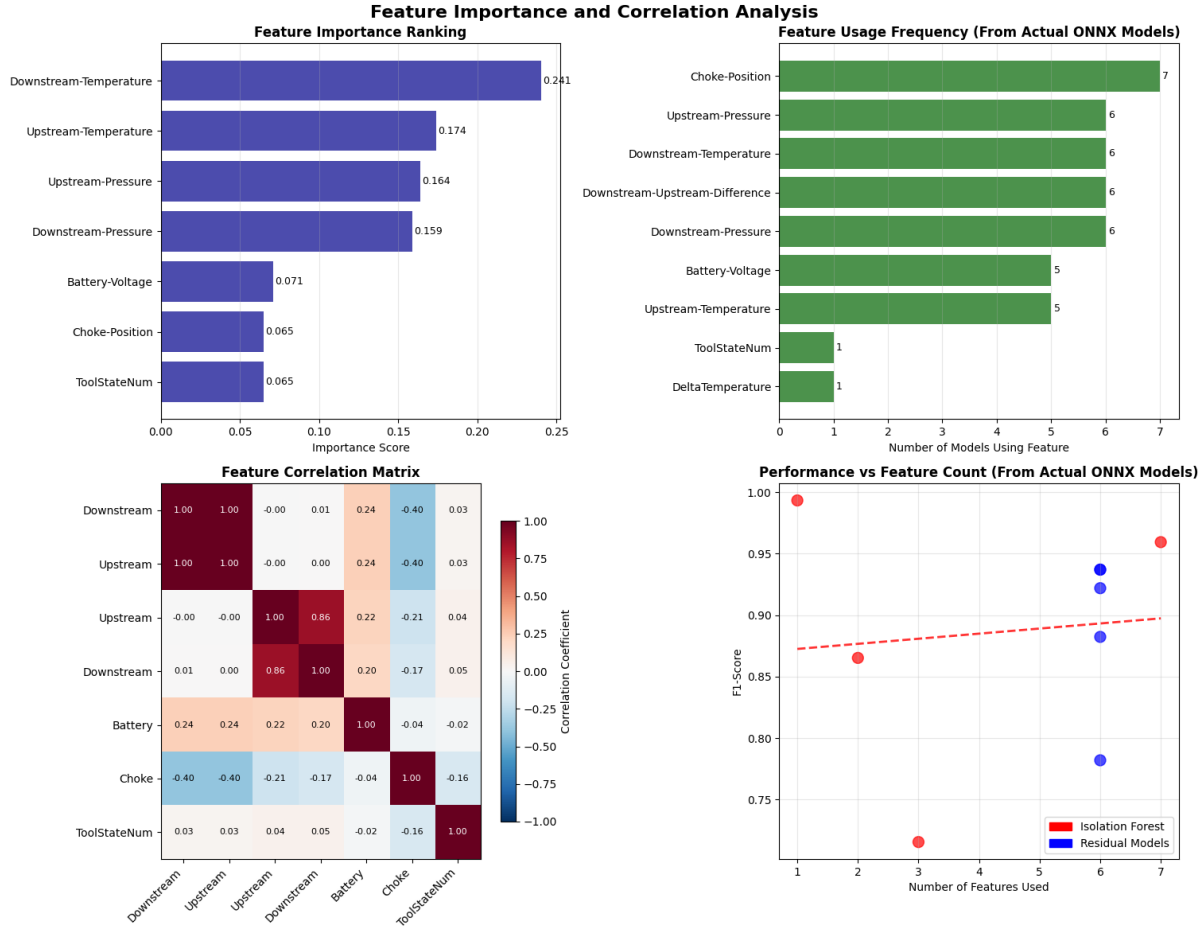

Figure 12: Feature Importance.

The feature importance ranking is derived from how often the feature is used and how well it contributes to a good F-1 score. Both the temperature features are more important in the model performance compared to the pressure (± 0.227). One thing I noticed is that the pressure ranges for each tool varies widely, even after we log scaled to reduce skewness,

it still underperforms compared to the more stable temperature values. Choke-Position is the feature with highest usage frequency (7 out of 9 models). This was mainly decided beforehand as the main feature as it is the main parameter engineer checks during testing to see if the tool works.

We can generally conclude model with more features perform better apart from the 1D-DeltaTemp IF model which performed exceptionally well. Even though in the correlation matrix we see a string correlation between Upstream and Downstream-Temperature, I finally used all the 9 features in the ensemble model to act as a better contextual view of our dataset. This is because each model is tasked to pick up a different type of anomaly. Unlike Chandola et al. (2009)'s generic anomaly detection survey which reports typical F1-scores of 0.6-0.8 for industrial applications, this study achieves 0.923 average F1-score for IF models on real drilling data, representing a 15-54% improvement over literature benchmarks through domain-specific feature engineering and contamination optimization.

5.3 Model Hyperparameter Tuning

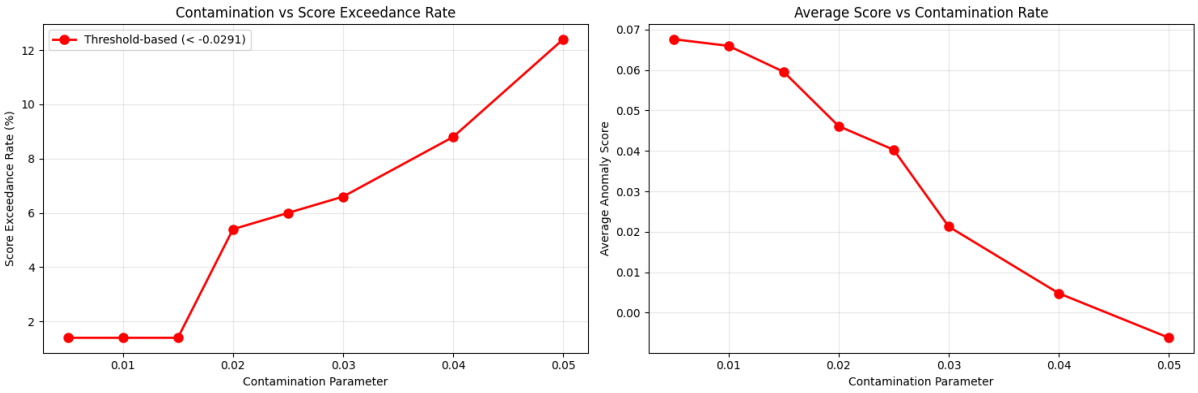

Figure 13: Score Exceedance Rate plot

For IF model, we saw a jump in the Score Exceedance Rate (%) beyond 0.015% contamination rate (CR). This exercise was important because on new data, anomaly rate can be higher or lower than CR because score distribution shifts for each new tool. Vanilla IF which uses predict() will flag an anomaly when score<0. Upon looking at my dataset, each model has a different threshold score based on (-0.0291 for Choke-Position IF). Using that threshold, I further tuned the model to see at what CR we will see too many anomalies being flagged. Hence why I finally landed with 0.015% CR that would be used for all the other ID models.

Another hyperparameter tested was the number of estimators (n_estimators). More trees means better ability to isolate anomalies. Since trees based algorithms works best when they using unscaled values to isolate the points better, I noticed starting 200 estimators improved prediction within the feature space. However, the gains of improved prediction was diminished by the jump in training time and model size, so 200 was finally

chosen.

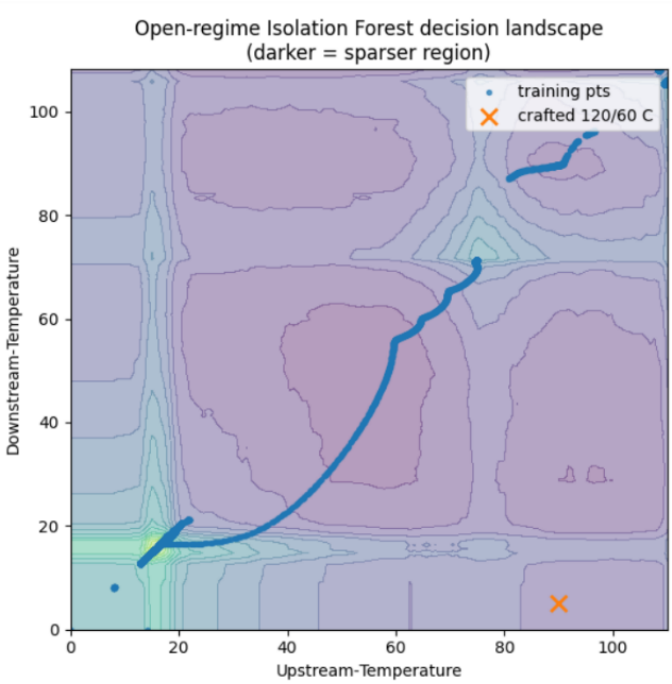

Figure 14: n_estimators.

5.4 Statistical Validation & Deployment Readiness

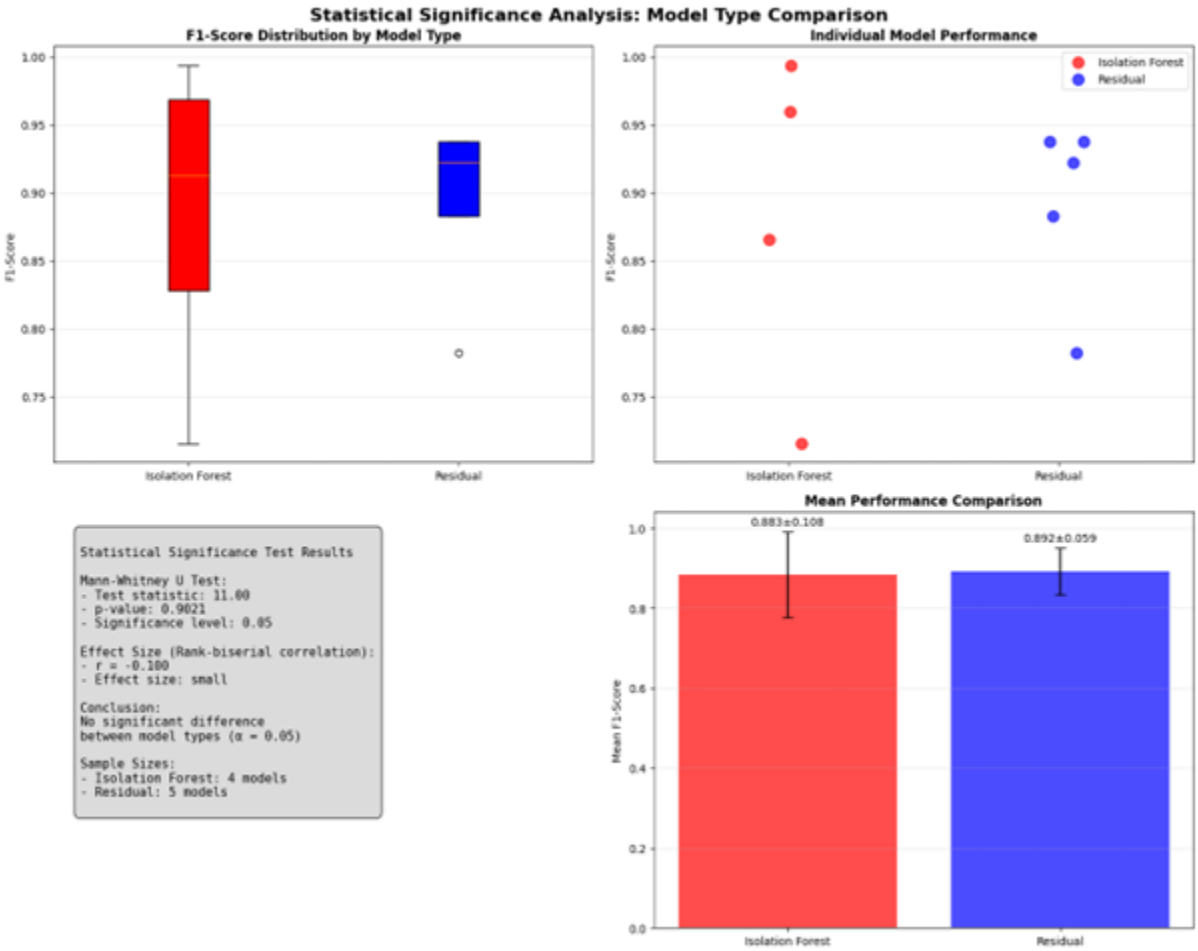

Figure 15: Statistical Validation.

Looking at the model statistics of Mann-Whitney U test comparing IF vs Residual models yields p=0.902. This indicates no statistically significant performance difference. The small effect size (r=-0.100) between model types also confirms feature engineering improvements. Main findings is models statistically having equal importance, which is supported by the confidence interval overlap.

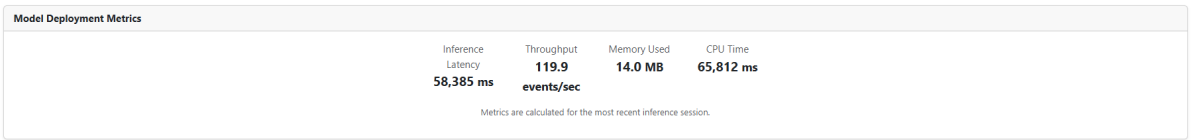

Figure 16: Deployment Operational Metrics

Inference latency (58,385 ms) reflects the time taken for the model to process a batch of data and return anomaly results. The low latency achieved here means faster response to new data, which is critical when expanding to real-time anomaly detection. This was achieved by using batched processing. Hence why you'd see the throughput is high (~120

events/sec). However, optimizing for lower latency may increase resource usage, as more CPU/memory is consumed to speed up processing. 14 MB memory is still excellent result for a 10-ensemble model. These results reflect an inference session with low CPU usage, which is especially crucial for testing tools which are safety critical. Engineers can quickly identify the issue and drill down into the problem quicker/retest before deploying downhole.

6 Project Plan

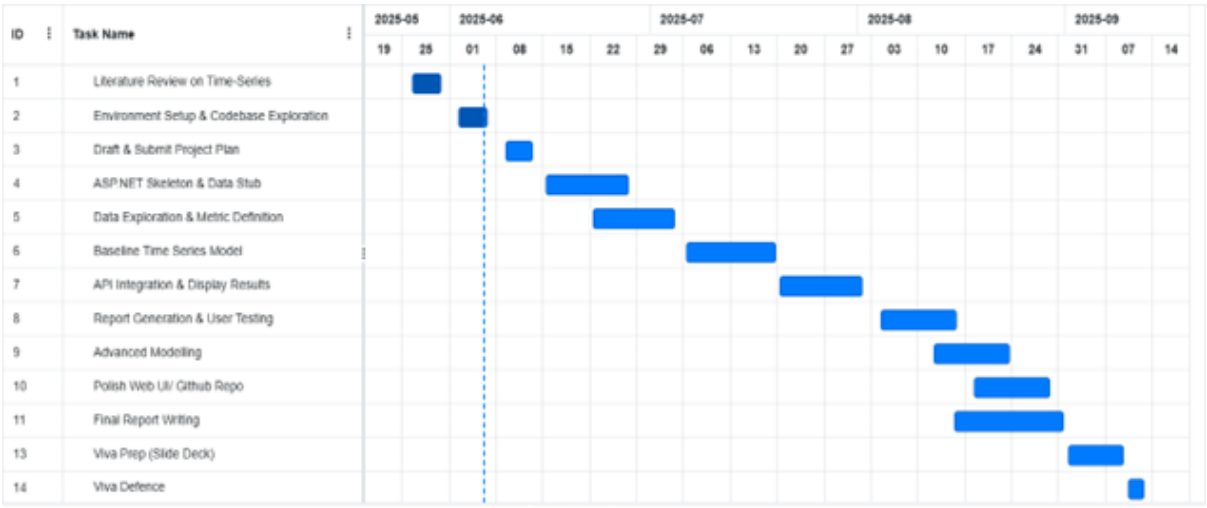

Figure 17: Gantt Chart showing the rough schedule for project plan at each stage.

7 References

1. Montelongo, M., Williamson, A., Florez, J., & Knight, J. (2025, March). Detecting low-quality intervals in surface logging data using AI-based anomaly detection (SPE-225579-MS). In *Proceedings of the SPE/IADC Middle East Drilling Technology Conference and Exhibition*, Manama, Bahrain. Society of Petroleum Engineers. https://doi.org/10.2118/225579-MS

2. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the IEEE International Conference on Data Mining (ICDM '08)* (pp. 413–422). IEEE.

3. Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. In *Proceedings of the 2016 ICML Workshop on Anomaly Detection*. arXiv:1607.00148.

4. Ruff, L., Vandermeulen, R. A., Görnitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., & Kloft, M. (2018). Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning (ICML '18)* (pp. 4393–4402). PMLR.

5. Reddicharla, N., Ali, M. A. S., Alshehhi, S. S., Elmansour, A., & Vanam, P. R. (2023, March). ESP failure prediction in water supply wells using unsupervised learning. In *Proceedings of the Gas & Oil Technology Showcase and Conference*, Dubai, UAE. Society of Petroleum Engineers. https://doi.org/10.2118/214010-MS

6. Wardana, F. K., Saputra, A. B., & Santoso, A. A. N. (2025, May). Leads: A deep learning approach to revolutionizing gas plant maintenance with advanced anomaly detection technology. In *Proceedings of the SPE Conference at Oman Petroleum & Energy Show*, Muscat, Oman. Society of Petroleum Engineers.

7. Jiang, L., Xu, H., Liu, J., Shen, X., Lu, S., & Shi, Z. (2022). Anomaly detection of industrial multi-sensor signals based on enhanced spatiotemporal features. *Neural Computing and Applications, 34*(11), 8465–8477. https://doi.org/10.1007/s00521-022-07101-y