ANOMALYZE FINAL DESIGN REVIEW REPORT

William Girdwood, John Greenough, Ella Kennedy, Manjary Muruganandan, Kiranjit Nagra
Team 9

Abstract – *HVAC systems are often maintained reactively, with faults detected only after performance degradation becomes visible. For Alert Labs, this creates a gap between continuous sensor data collection and actionable maintenance insight. This project develops Anomalyze, a hybrid fault detection and decision-support system that combines anomaly detection, supervised learning, and an interactive diagnostic dashboard. The solution supports early fault identification, system-level monitoring, and unit-level investigation. Evaluation through core-based validation, preprocessing analysis, and usability testing demonstrated improved interpretability and stronger support for operational decision-making.*

Keywords: HVAC fault detection, time-series machine learning, anomaly detection, predictive maintenance, diagnostic dashboard

1. PROBLEM ANALYSIS

Heating, Ventilation, and Air Conditioning (HVAC) systems are critical infrastructure across residential, commercial, and industrial buildings, responsible for thermal comfort and indoor air quality. In Ontario, approximately 75% of homes rely on forced-air furnaces and 85% have access to air conditioning (Statistics Canada, 2023). Despite this prevalence, HVAC systems are still largely maintained using reactive or schedule-based strategies, where intervention occurs only after performance degradation or failure becomes visible (Alsaleem et al., 2014). A 2024 Ontario survey found that 76% of homeowners experienced at least one home repair emergency in a single year, and HVAC systems accounted for 26% of those incidents. (Service Line Warranties of Canada, 2024). These figures show that HVAC faults are not isolated events, but a frequent and costly operational issue.

1.1. Industry Partner

This broader industry problem is especially relevant for Alert Labs, an HVAC monitoring and service provider that collects continuous sensor data across a large portfolio of systems. Its central challenge is not data collection, but converting that data into timely and actionable maintenance insight. As shown in Figure 1, AlertLabs deploys a range of IoT-enabled devices to capture system-level signals such as temperature, pressure, and flow across different environments. While IoT-enabled devices have greatly expanded data availability, they have not solved the harder problem of identifying early signs of failure and determining when intervention is needed (Chen et al., 2023). As a result, despite advances in Fault Detection and Diagnostics (FDD), organizations like Alert Labs can remain data-rich but insight-poor.

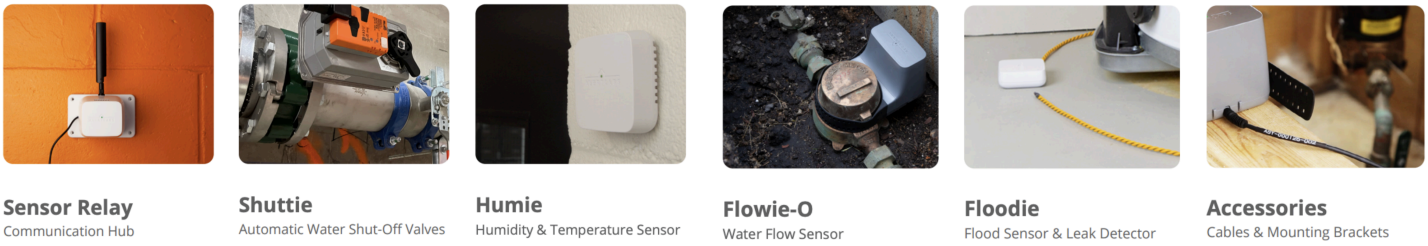


Figure 1: AlertLab Products

1.2. Causes of the Problem

Several interacting factors make early fault detection difficult for Alert Labs. First, many HVAC faults develop gradually and are difficult to distinguish from normal system variability. Issues such as refrigerant leaks, airflow restrictions, and component wear may remain within nominal ranges in their early stages, allowing systems to operate inefficiently before conventional alerts are triggered. Soft faults can increase energy consumption by 15–30% before detection, highlighting the cost of delayed intervention (Chen et al., 2023). A classification of HVAC pump failure causes is shown in Figure 2.

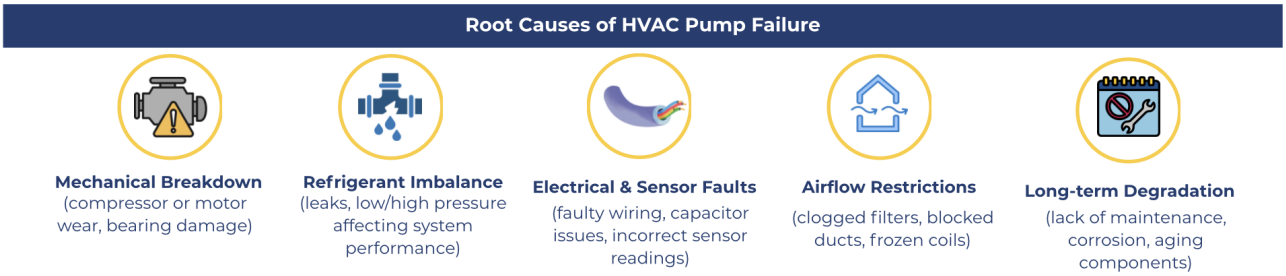


Figure 2: Root Causes of HVAC Pump Failure

Second, the data collected by Alert Labs are inherently high-dimensional, noisy, and time-dependent. Detecting faults requires identifying patterns across multiple sensors over time, rather than relying on single abnormal readings (Bi et al., 2024). This creates substantial analytical complexity, since useful signals must be extracted through time-series analysis such as rolling trends, cross-sensor relationships, and deviations from expected behaviour. A further challenge is that much of Alert Labs' data is not paired with reliable fault-time labels. While continuous monitoring produces large volumes of operational data, the exact onset and progression of faults are often not explicitly recorded, limiting the effectiveness of purely supervised learning approaches.

Third, variability across HVAC systems reduces the usefulness of fixed thresholds and rule-based methods. Differences in configuration, operating environment, and maintenance history mean that thresholds that are meaningful for one unit may be misleading for another, increasing the risk of missed faults and false alarms (Chen et al., 2023). This challenge is further amplified by the lack of standardized open datasets and reproducible benchmarks.

Finally, there is a usability challenge. Even when models achieve strong technical performance, their outputs may not support operational decision-making. For Alert Labs, value depends not only on detecting abnormal behaviour, but also on comparing risk across systems, identifying contributing signals, and determining when intervention is warranted. When outputs are not interpretable or actionable, technical performance does not easily translate into maintenance value.

1.3. Stakeholders and Impact

The impact of this problem is multi-layered, with Alert Labs at the center. Building owners and occupants experience unexpected failures, comfort disruption, and higher utility costs caused by undetected efficiency degradation (Chen et al., 2023). Maintenance teams face reactive workload spikes and limited prioritization support when managing large equipment portfolios. For Alert Labs, the impact is both operational and strategic: without reliable and interpretable fault detection, it is difficult to prioritize at-risk units, reduce unnecessary service calls, and demonstrate value to customers. More broadly, undetected HVAC inefficiencies also carry cost and sustainability implications.

1.4. Existing Solutions and Limitations

Existing approaches to HVAC fault detection do not fully address Alert Labs' needs. Rule-based threshold monitoring is widely used due to its simplicity, but it is static and poorly suited to gradual faults and system variability. These systems often miss early-stage degradation and can generate false positives that reduce trust over time (Chen et al., 2023) Physics-based models improve diagnostic accuracy by comparing observed and expected system behaviour, but require system-specific calibration and are difficult to scale across diverse equipment fleets (Chen et al., 2023) Supervised machine learning approaches, including Random Forest and deep learning models, demonstrate strong performance on

labelled datasets (Bi et al., 2024) but depend on high-quality fault labels that are rarely available in real-world HVAC deployments. Unsupervised and hybrid anomaly detection methods address the label limitation but often produce high false-positive rates when used independently, making them difficult to rely on for operational decision-making (Paolini et al., 2026). While explainability-focused methods are emerging (Park et al., 2026), they are not yet widely integrated into scalable, trusted, user-facing monitoring systems.

1.5. Problem Statement

For Alert Labs, the core gap is the lack of an interpretable decision-support system that can translate continuous HVAC sensor data into earlier, more usable maintenance insight. Although sensor data are already collected, limited fault-time labels constrain supervised modelling, and anomaly signals alone are difficult to interpret operationally. As shown in Figure 3, failures are often preceded by hidden degradation and missed early warning signals. This motivates a hybrid fault detection approach and diagnostic dashboard that support earlier fault identification, risk prioritization, and investigation across systems.

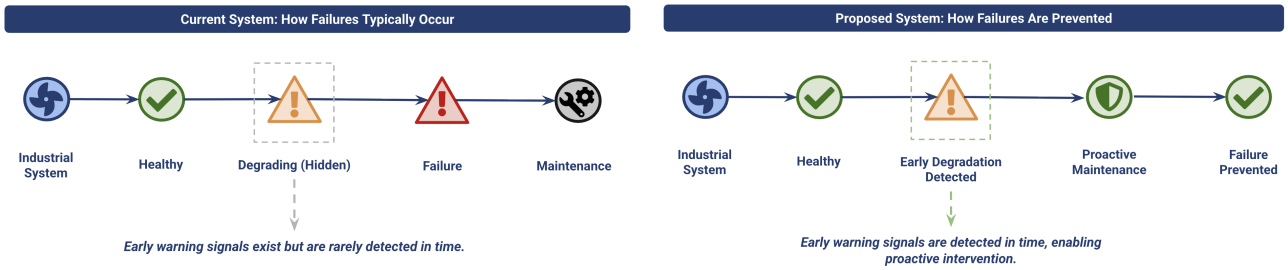


Figure 3: Current vs Proposed Failure Detection

2. REQUIREMENTS ANALYSIS

System requirements were developed through an iterative, stakeholder-informed process grounded in problem analysis and technical domain expertise. Initial guidance was provided by Chirag Seth, a PhD researcher who had direct involvement in the underlying model development and the technical objectives of the Alert Labs system. Because he was closely familiar with both the modelling approach and the operational goals of the project, he was well positioned to define the core system capabilities required for early fault detection, interpretable outputs, and practical usability. These requirements were initially communicated verbally and were then translated by the team into a structured framework of functional, non-functional, model-specific, and UI/UX requirements. The final requirements are directly traceable to the core problem of delayed fault detection and limited interpretability. The full specification is provided in Appendix A.

2.1. Functional Requirements

Functional requirements define the system's ability to ingest standardized .csv time-series data, store historical datasets, and perform fault analysis using supervised classification and unsupervised anomaly detection. The system must also support simulated early fault flagging and present results through an interpretable dashboard for both single-unit and fleet-level analysis.

2.2. Non Functional Requirements

Non-functional requirements focus on usability, performance, and maintainability. The system must support straightforward data upload, return results within approximately 20 seconds, and remain clear and usable for stakeholders. Reproducibility, modularity, and ease of handoff were also prioritized to support validation and deployment.

2.3. Model Specific Requirements

Model-specific requirements focus on reliable fault classification and early detection under noisy, time-dependent conditions. A shared feature set is maintained across models for consistency, and performance is evaluated using standard metrics with a target F1-score of at least 80%.

2.4. UI/UX Requirements

UI/UX requirements focus on making outputs interpretable, interactive, and accessible to non-technical users. These include clear visual differentiation of system states, support for exploration through zooming and filtering, and consistent presentation of model outputs and performance metrics.

3. SOLUTION DEVELOPMENT

3.1 System Overview

The solution pipeline transforms raw HVAC sensor data into fault classification insights using machine learning, identifying early fault patterns within high-dimensional time-series data and translating them into actionable maintenance insights. The pipeline consists of four stages: data preprocessing, feature engineering, model development, and fault classification, followed by visualization in an interactive monitoring interface. It is structured as a two-part architecture, where the Fault Detection Engine performs anomaly detection and classification to generate fault probabilities, and the Monitoring and Diagnostic Interface presents these outputs to support system-level monitoring and unit-level investigation.

3.2 Part 1: Fault Detection Engine

3.2.1 Summary

The overall fault detection pipeline is illustrated in Figure 4, outlining how raw HVAC sensor data is transformed through preprocessing, feature engineering, and hybrid model-based classification into fault predictions.

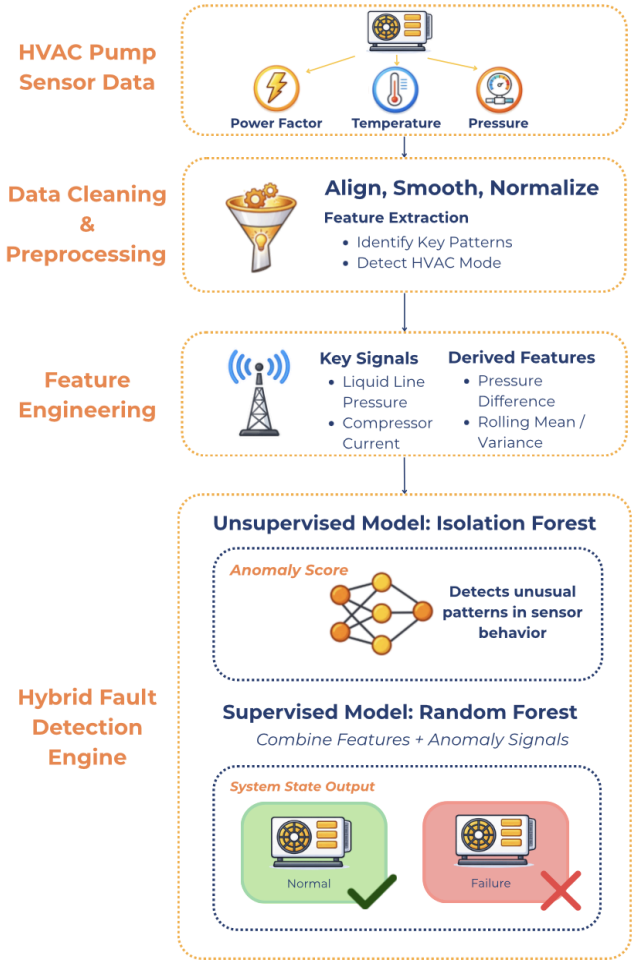


Figure 4: Pipeline for Fault Detection Engine

3.2.2 Data Summary & Preprocessing

The dataset consists of time-series HVAC sensor data collected across multiple cores over approximately nine months, totaling over 5.5 million observations. Key features include pressure, temperature, and electrical signals (e.g., compressor current), capturing system behaviour under both normal and faulty conditions. To prepare the data for modelling, preprocessing focused on ensuring consistency and usability. Non-informative features (e.g., high missingness or low variance) were removed, and records were aligned chronologically per core to preserve temporal structure. Noise in sensor readings was addressed through basic smoothing and filtering, given the high variability in normal HVAC operation. Features were standardized to ensure comparable scale across variables, and the dataset was labeled at the core level as normal or faulty to support supervised learning, acknowledging that exact fault timestamps were not available.

3.2.3 Feature Engineering

Following preprocessing, feature engineering introduced additional signals that capture system behaviour beyond raw sensor readings. In addition to the original features (e.g., pressure, temperature, and current), derived features were introduced to better represent operating conditions and deviations from normal behaviour. These included encoded system states (e.g., HVAC mode, converted into categorical indicators) and an anomaly score generated using Isolation Forest, which assigns a higher value to observations that deviate from typical system operation.

To capture temporal dynamics, rolling statistics such as short-term averages and variability measures were optionally included to reflect trends and fluctuations over time. Feature relevance was assessed post-training using Random Forest feature importance and correlation with both the fault label and predicted probabilities. As shown in Figure 5, several pressure and temperature related features exhibit strong correlation with predicted failure probability, highlighting their importance in driving model predictions. Features that consistently contributed to model decisions were retained, while unstable or low-value features were removed to improve model robustness and interpretability.

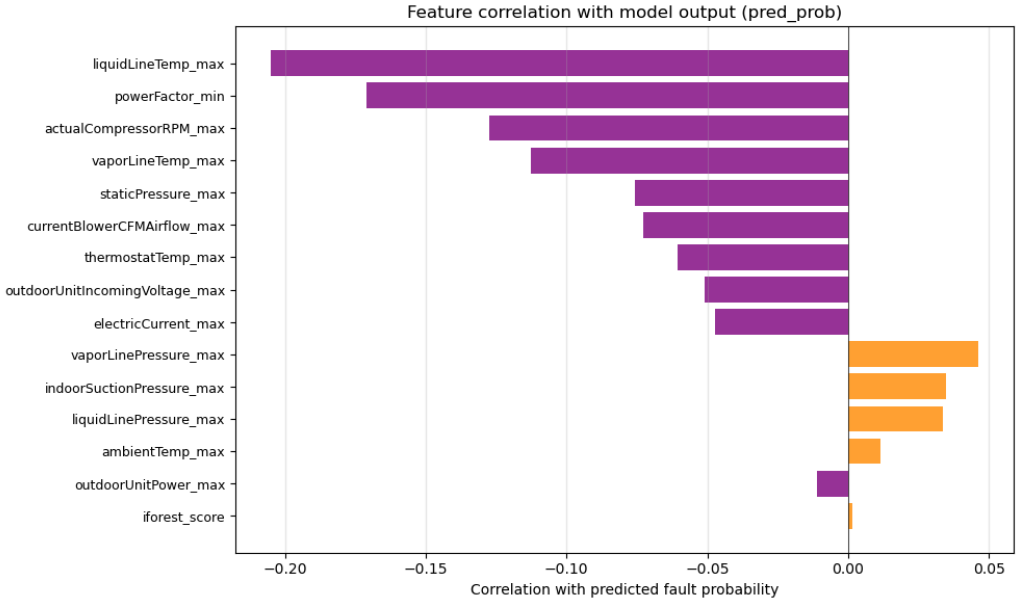


Figure 5: Top Features associated with Predicted Failure Probability

3.2.4 Hybrid Fault Detection Engine

The fault detection engine uses a hybrid modelling approach that combines unsupervised anomaly detection with supervised classification to address noisy data and limited label precision. An Isolation Forest is first applied to assign an anomaly score to each observation based on its isolation within the feature space, capturing deviations from typical system behaviour without relying on labeled data. This anomaly score is then used alongside the original features as input to a Random Forest classifier, which predicts whether the system is operating under normal or faulty conditions.

Several modelling approaches were evaluated, each presenting distinct trade-offs. Random Forest provided strong classification performance and interpretability but depended on coarse system-level labels, limiting its ability to detect early-stage faults. Isolation Forest was effective at identifying anomalous behaviour without labels, but when used independently, it produced a high number of false positives and lacked decision boundaries for actionable classification. Sequence-based models such as LSTM were explored to capture temporal dependencies; however, they introduced increased model complexity, longer training times, and reduced interpretability, making them less suitable for deployment in an operational setting.

The hybrid Isolation Forest + Random Forest approach was selected as it balances these trade-offs. It leverages Isolation Forest to capture early deviations in system behaviour, while Random Forest filters these signals into more stable and interpretable fault predictions. As shown in Table 1, this approach achieved the strongest overall performance, with high accuracy (0.978), precision (0.974), recall (0.816), and ROC AUC (0.888). This reflects a balance between sensitivity to early faults and robustness to false positives, which is critical for operational use. The hybrid design is therefore well suited to the dataset, where faults develop gradually and labels are limited to system-level annotations.

	Accuracy	Precision	Recall	ROC AUC
Random Forest (RF) (Balanced)	0.843	0.870	0.775	0.821
RF (Recall-Focused)	0.750	0.762	0.626	0.532
RF (Precision-Focused)	0.654	0.994	0.034	0.067
LSTM Autoencoder	0.647	0.536	0.769	0.557
Isolation Forest + RF (Hybrid)	**0.978**	**0.974**	**0.816**	0.888

Table 1: Model Results

Disclaimer: The machine learning models and baseline results presented here were developed by Chirag Seth and are used in this project for evaluation and interface design purposes.

3.3 Part 2: Monitoring & Diagnostic Interface

3.3.1 Summary

To complement the fault detection engine, a Monitoring and Diagnostic Interface was developed to translate model outputs into actionable insights for engineers. While the model generates fault probabilities, the interface enables interpretation, prioritization, and investigation of system risk. It is structured into three components: a homepage with configuration controls, a global overview for system-level monitoring, and a single-core view for detailed diagnostics. This structure aligns with an operational workflow of monitor → prioritize → investigate, ensuring that model outputs directly support decision-making. As illustrated in Figure 6, the interface integrates system-level summaries with detailed diagnostic views to support both high-level monitoring and in-depth analysis.

The interface was implemented in Streamlit because it enabled rapid development of an interactive, Python-based dashboard that could be tightly integrated with the machine learning pipeline and iterated quickly within the project timeline. This made it well suited for the requirements, where usability testing and interface refinement were as important as deployment speed. With additional time, a more production-oriented web framework could have been considered to support stronger real-time data ingestion, multi-user deployment, and tighter integration with an operational environment.

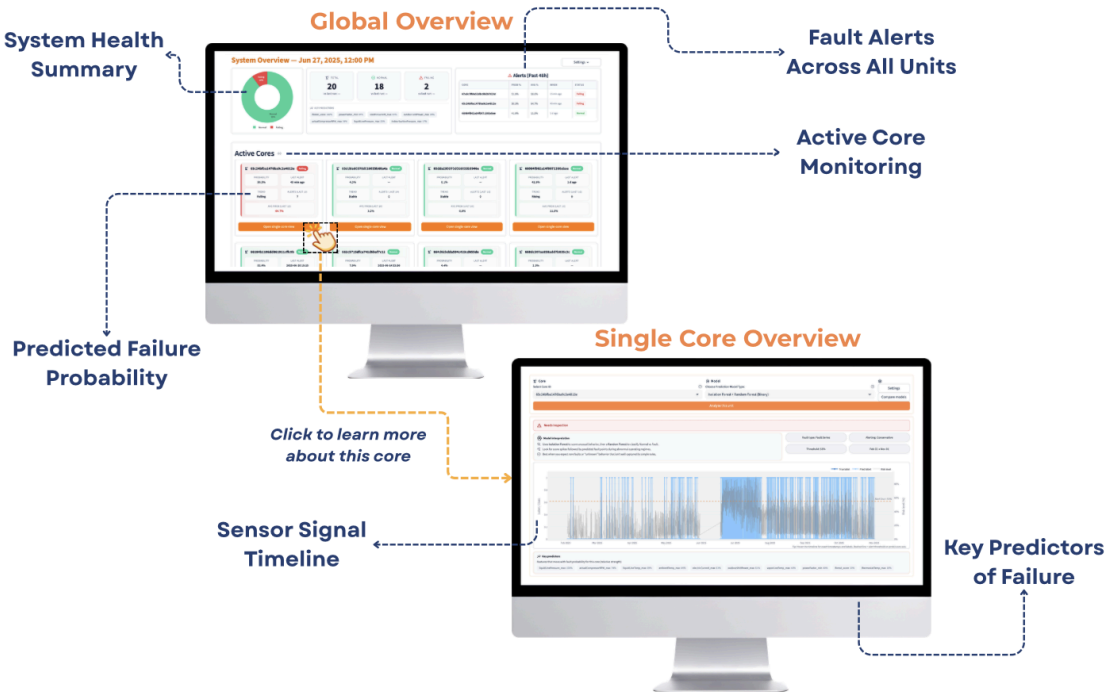


Figure 6: Overview of Interface Main Pages & Key Features

3.3.2 System-Level Monitoring

The global overview provides a consolidated view of all monitored HVAC cores, enabling rapid assessment of overall system health and prioritization of maintenance actions. It aggregates model outputs into key indicators, including total units, system status distribution (normal vs. failing), and recent alerts. Each core is associated with its predicted fault probability and current status, allowing engineers to quickly identify high-risk systems without inspecting individual units, as shown in Figure 7.

The interface supports decision-making by enabling comparison and ranking of cores based on risk, while also surfacing recent alerts and key contributing features. This reduces manual inspection effort and allows engineers to focus on the most critical systems first. By presenting model outputs in a structured and interpretable format, the global overview directly supports efficient triage and system-level monitoring in environments with many assets.

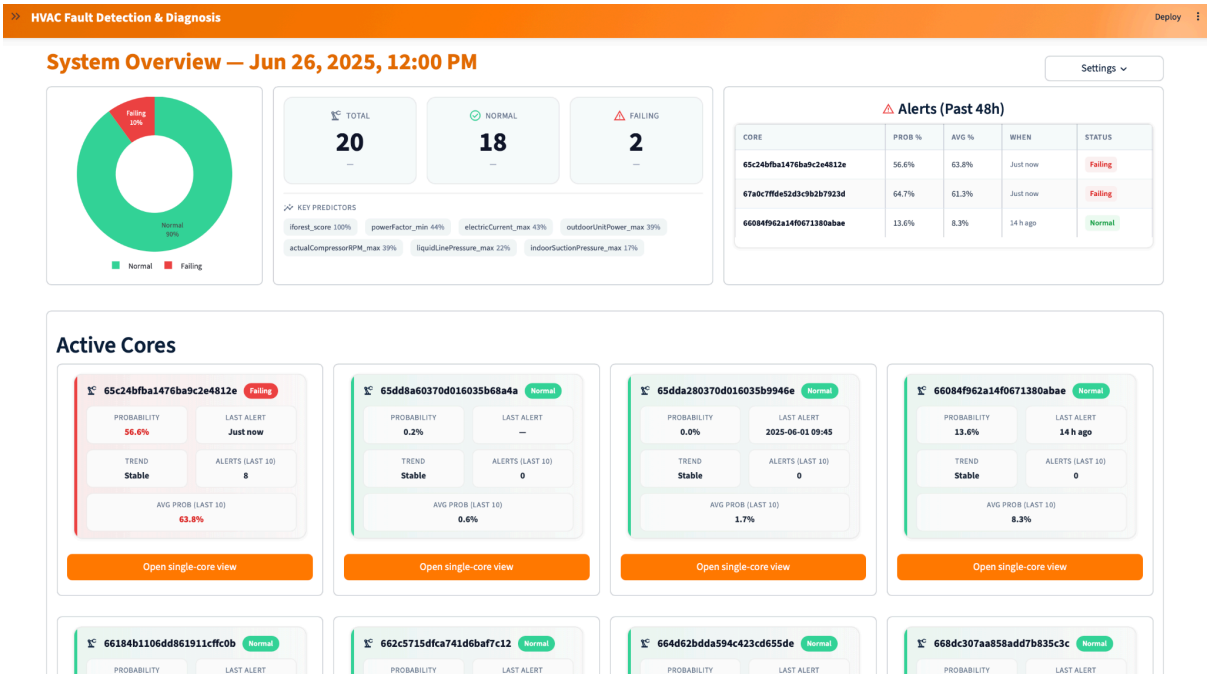


Figure 7: Global Overview Page

3.3.3 Unit-Level Diagnostics

From the global overview, users can select an individual core to access a detailed diagnostic view, enabling deeper investigation of system behaviour over time. This view integrates model outputs with raw sensor data, displaying fault probabilities, alert thresholds, and time-series signals in a unified timeline, as shown in Figure 8. By visualizing how predicted risk evolves relative to sensor patterns, engineers can assess whether detected anomalies correspond to meaningful changes in system operation.

Additional diagnostic features support interpretation and validation of model outputs. These include recent alert logs, probability distributions, and key contributing features, as well as performance metrics such as precision, recall, and F1-score. The interface also allows comparison across models and inspection of anomaly signals alongside sensor readings, providing transparency into how predictions are generated. Together, these features enable engineers to move beyond binary alerts, supporting root-cause analysis, validation of model behaviour, and more informed maintenance decisions.

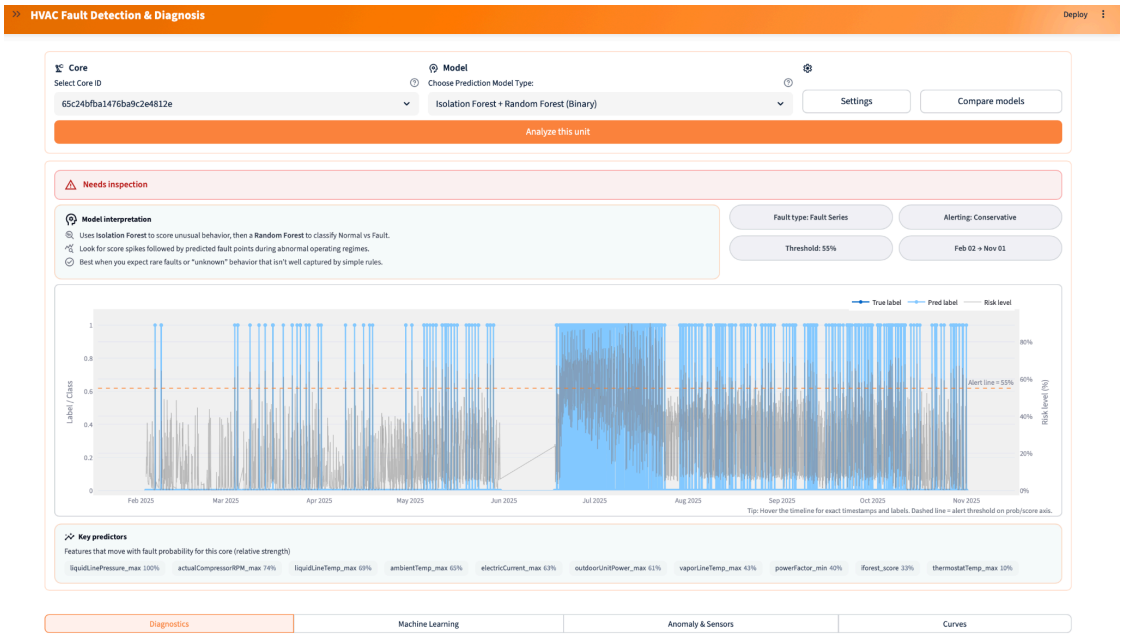


Figure 8: Single Core Page

3.3.3 Deployment

The current system is deployed using a private Streamlit Cloud deployment and is structured around manual .csv file upload rather than live data ingestion. This makes it suitable for controlled diagnostic use, model interpretation, and validation, but limits real-time alerting, automated integration, and evaluation in a fully operational environment. The project will be continued by PhD student Chirag Seth, and Alert Labs intends to use the interface in partnership with him as a diagnostic tool to better understand, validate, and refine the model outputs prior to broader deployment. With a longer development timeframe, the system could be extended to support continuous monitoring, a feedback loop for threshold refinement and user validation, and migration to a more production-oriented framework better suited to live deployment.

4. SOLUTION EVALUATION

The system was evaluated using both validation and verification. Validation assessed predictive performance using labeled data and core-based holdout testing, with precision, recall, and F1-score used to measure generalization under class imbalance. Alternative models, including Random Forest and LSTM, were also considered (Section 3.2.4). Verification assessed whether outputs were physically and operationally meaningful, using liquid-line pressure comparisons and interface-based review against sensor behaviour. Together, these methods evaluate both statistical performance and practical credibility.

4.1 UI/UX Evaluation and Iterative Redesign

The Anomalyze interface was refined using an iterative User-Centred Design (UCD) process combining think-aloud testing (n = 6), expert review, redesign, and post-redesign survey evaluation (n = 11) with stakeholder validation. This multi-method process identified usability issues, aligned the interface with real workflows, and quantified improvements.

4.1.1 Initial Usability Findings

The initial think-aloud study revealed three primary usability issues. First, users demonstrated poor homepage comprehension, frequently expressing uncertainty regarding system purpose (e.g., “I’m not really sure what I’m supposed to do here”). Second, workflow structure was unclear, as users were unable to determine the correct sequence of actions (e.g., “Do I pick a model first or look at the data?”). Third, users experienced high cognitive load due to excessive co-visible information and an unclear distinction between system-level monitoring and unit-level diagnostics. These findings indicate failures in mental model formation and violations of information hierarchy principles. Expert review (Chirag Seth) confirmed that the interface was misaligned with real-world operational workflows, which follow a monitor → prioritize → investigate sequence. Additionally, model outputs were identified as non-actionable, as fault probabilities were presented without thresholds or contextual explanation. As noted in the review, “there’s no way to tell if this score is

actually concerning or just noise." As a result, the system functioned as an analytical tool rather than a decision-support system.

4.1.2 Redesign Approach

The interface was redesigned using a progressive disclosure approach and restructured into two hierarchical views: a Global Overview for monitoring and prioritization, and a Single-Core Deep Dive for diagnostic investigation. Key improvements included the introduction of colour-coded unit cards for rapid triage, threshold-labelled probability timelines and explicit alert states (e.g., "Needs inspection"), feature-level explanations (Key Predictors), and sensor-level overlays to support validation of model outputs. These changes reduced cognitive load, clarified navigation, and improved the actionability of model outputs. A full redesign framework can be found in Appendix C.

4.1.3 Post-Redesign Results

Post-redesign evaluation results (n = 11) demonstrated significant usability improvements. Overall, 82% of participants reported being able to use the system without significant difficulty. The Global Overview performed strongest, with over 85% of users correctly identifying critical units. Workflow clarity showed the largest improvement, although it remained the primary residual issue. Minor ambiguity in homepage comprehension persisted for two participants, and one participant misinterpreted the risk score as a measure of model accuracy, indicating a need for clearer labeling. A stakeholder demonstration with Alert Labs engineers provided external validation of the redesigned interface. Engineers confirmed improvements in triage efficiency and interpretability, particularly highlighting the ability to zoom into specific timestamps and overlay sensor data with model outputs. These results indicate that the interface now better supports operational decision-making and aligns with real-world workflows.

4.2 Model Evaluation

4.2.1 Scaling Methods

Feature scaling was evaluated because it directly affects how sensor variables and downstream anomaly scores are represented before entering the Isolation Forest and Random Forest components of the hybrid pipeline. The choice of scaler, as well as whether scaling is performed globally or separately for each core, changes the geometric structure of the input space, the stability of per-core anomaly detection, and the extent to which a single deployment threshold can generalize across different systems. For this reason, a structured comparison was necessary to support a defensible preprocessing choice rather than relying on an ad hoc configuration that might perform well on one data split but fail on unseen cores.

The hybrid IF + RF pipeline was evaluated under six configurations: StandardScaler, MinMaxScaler, and RobustScaler, each with global and local fitting. Generalization was assessed using 5-fold core-based holdout validation, in which a subset of cores was excluded entirely from training in each fold and used only for evaluation. Performance was summarized at the unit level using precision, recall, and F1, with F1 emphasized due to class imbalance.

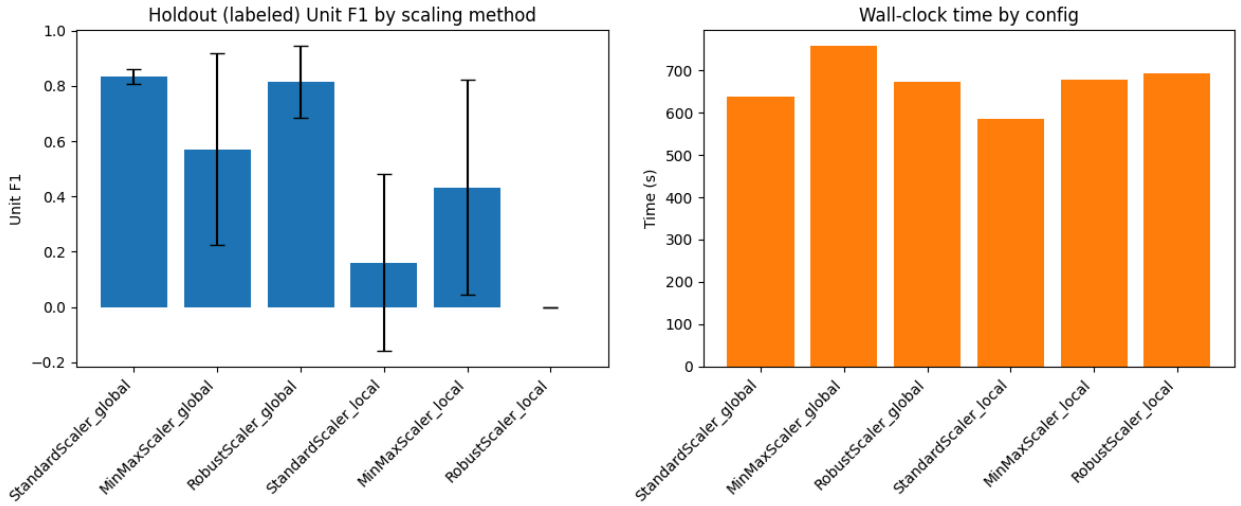


Figure 9: F1 and Time Comparison of Scaling Methods

As shown in Figure 9, global StandardScaler achieved the highest mean holdout F1 with relatively low variability across runs. MinMaxScaler performed worse and was less stable, while RobustScaler was intermediate. Local scaling configurations were consistently weaker, suggesting reduced generalization across unseen cores. These results support global StandardScaler as the preferred preprocessing strategy, with scaler choice and deployment threshold interpreted jointly.

4.2.2 Manual Liquid Line Analysis

Liquid-line pressure was used as a qualitative, physics-informed verification check when reviewing model behaviour on training and external test time series. In HVAC systems, elevated liquid-line pressure can indicate that the system is working harder under abnormal conditions such as restrictions, blockages, or leakage-related inefficiencies. It was therefore used as a simple proxy for physical stress.

The purpose of this analysis was verification rather than formal validation. Liquid-line pressure timelines, and where applicable heuristic high-pressure flags, were compared visually against model risk scores to assess whether periods of elevated physical stress tended to coincide with increased predicted fault risk. This served as a useful sanity check that the pipeline was responsive to meaningful sensor behaviour and that preprocessing, scaling, and threshold choices did not produce outputs that were flat or disconnected from a basic physical indicator of system strain. As shown in Figure 10, the faulted core exhibits periods where higher liquid-line pressure coincides with elevated or more volatile model fault probabilities, suggesting that the model is responsive to physically plausible stress behaviour. In contrast, Figure 11 shows a normal core, where model probabilities remain comparatively low and stable despite ordinary variation in liquid-line pressure. Together, these figures provide a useful sanity check that the pipeline is not producing outputs that are flat, erratic, or disconnected from an interpretable system signal.

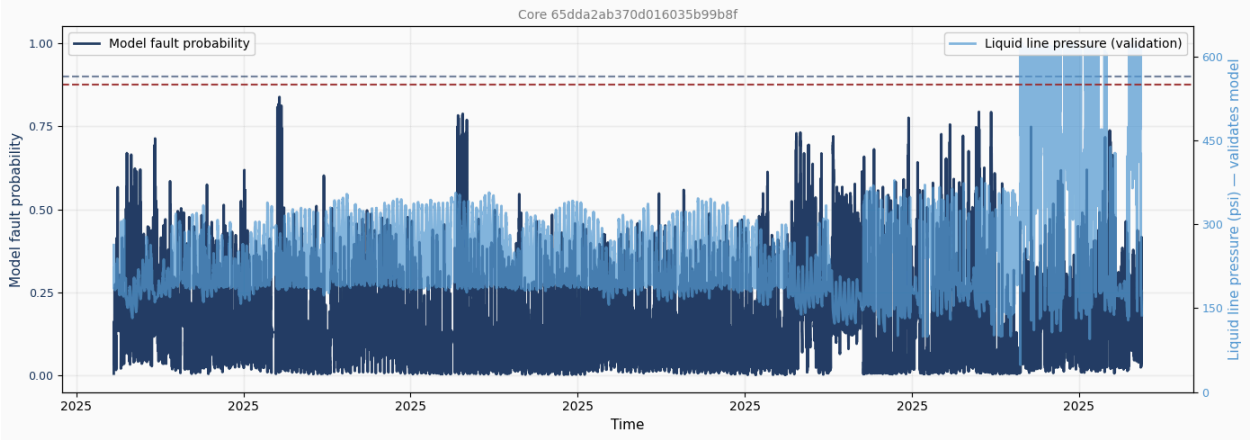


Figure 10: Faulted core behaviour: model fault probability and liquid-line pressure over time.

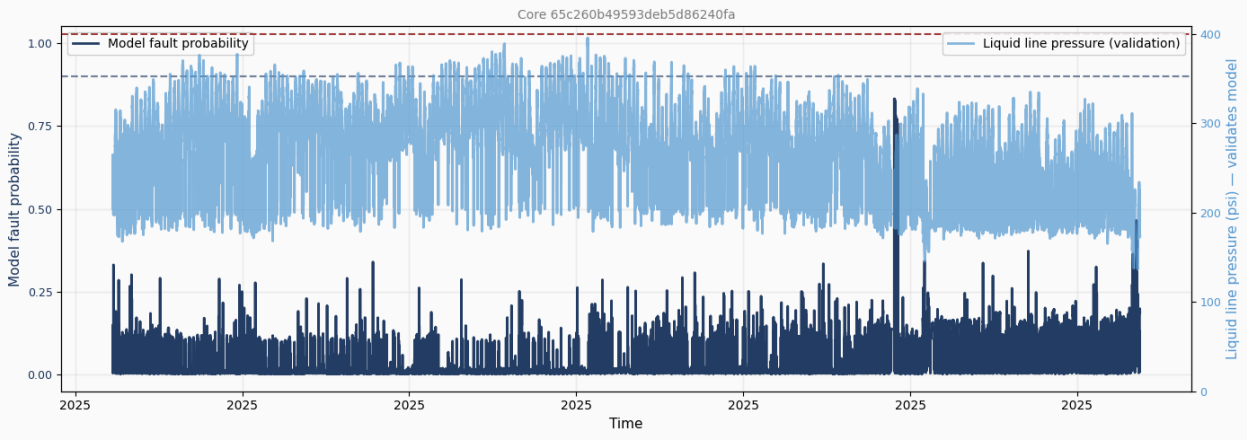


Figure 11: Normal core behaviour: model fault probability and liquid-line pressure over time.

This analysis also has clear limitations. High liquid-line pressure is not equivalent to the project's fault label and cannot be treated as ground truth. Some faults may not appear through elevated pressure, and high pressure may also occur without a true fault. Accordingly, this comparison was not used to select models, scalers, or deployment thresholds.

4.3 Limitations

While the proposed solution demonstrates strong performance in detecting early signs of HVAC system faults, several limitations were identified during development. These limitations primarily relate to data constraints, model assumptions, and deployment considerations, and highlight opportunities for future improvement.

One key limitation is the availability and granularity of labeled data. Fault labels were provided at the core level, indicating whether a system experienced a fault within a given period, but not the exact timestamp at which the fault began. As a result, the model is trained to recognize general patterns associated with faulty behavior rather than precise fault onset. This limits the ability to rigorously evaluate early detection performance at a fine temporal resolution and introduces uncertainty when interpreting predictions at specific time points. Another limitation is the generalization of the model across diverse HVAC systems. While the use of global feature scaling and a unified model supports consistency across systems, variations in system configuration, usage patterns, and environmental conditions may impact model performance in unseen scenarios. Additional validation across a broader range of systems would be required to ensure robustness in real-world deployment.

From a modelling perspective, the current approach focuses on binary classification of system state as normal or faulty. This simplifies the problem but does not distinguish between different types of faults, which may have distinct causes and require different maintenance actions. Expanding the model to support multi-class classification or fault-type identification would enhance its practical utility for engineers. Finally, while the interface provides interpretable outputs and supports investigation, the system has not yet been fully integrated into a live operational environment. As a result, threshold selection and alerting strategies are based on offline validation rather than real-time feedback. Deployment in a production setting would enable continuous calibration of model thresholds, incorporation of user feedback, and further refinement of the system's decision-making capabilities.

5. ACKNOWLEDGEMENTS

The team would like to thank Professor Mehrdad Pirnia for his guidance throughout the term. We also thank Chirag Seth for his contributions to model development and mentorship, as well as the Alert Labs team, especially Charlie Harland and Max Pfeifle, for their feedback and guidance. Finally, we thank Professor Ada Hurst, Chris Rennick, and the MSE 401/402 TAs for their support with course deliverables and requirements.

6. REFERENCES

Alsaleem, F., Abiprojo, R., Arensmeier, J., & Hemmelgarn, G. (2014). *HVAC System Cloud Based Diagnostics Model*. https://engineering.purdue.edu/HVACFDD/pdfs/Workshop_on_FDD_for_RTUs_Presentations/HVAC_Cloud-Based_Diagnostics-Alsaleem.pdf

Llopis-Mengual Belén, Yuill, D. P., & Navarro-Peris, E. (2025). Time series analysis of field data for soft faults detection and degradation assessment in residential air conditioning systems. *Applied Thermal Engineering*, *169*, 126104. https://doi.org/10.1016/j.applthermaleng.2025.126104

Bi, J., Wang, H., Yan, E., Wang, C., Yan, K., Jiang, L., & Yang, B. (2024). AI in HVAC fault detection and diagnosis: A systematic review. *Energy Reviews*, *3*(2), 100071–100071. https://doi.org/10.1016/j.enrev.2024.100071

Chen, Z., Zheng O'Neill, Wen, J., Pradhan, O., Yang, T., Lu, X., Lin, G., Miyata, S., Lee, S., Shen, C., Chiosa, R., Marco Savino Piscitelli, Capozzoli, A., Hengel, F., Kührer, A., Pritoni, M., Liu, W., Clauß, J., Chen, Y., & Herr, T. (2023). A review of data-driven fault detection and diagnostics for building HVAC systems. *Applied Energy*, *339*, 121030–121030. https://doi.org/10.1016/j.apenergy.2023.121030

Dang, H.-L., & Kwak, S. (2020). Review of Health Monitoring Techniques for Capacitors Used in Power Electronics Converters. *Sensors*, *20*(13), 3740. https://doi.org/10.3390/s20133740

Gao, Y., Hu, Z., Otomo, J., & Ke, Y. (2026). Contrastive self-supervised learning for lightweight and automated fault detection and diagnosis in HVAC systems. *Applied Energy*, *410*, 127557. https://doi.org/10.1016/j.apenergy.2026.127557

Greenough, J. (2026, March 9). Capstone peer critique – User testing [Data set]. Qualtrics. https://uwaterloo.yul1.qualtrics.com/reporting-dashboard/web/69b2d814b8413f000897e743/pages/Page_3939941e-8bdb-46ac-85fc-b45ac1ecf2dc/view?surveyId=SV_39NrRwMMdFh9x4O

Guo, W., Qiu, A., & Li, X. (2019). Active Fault Tolerant Control for HVAC System Based on GIMC and Feedforward Compensation. *2019 CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*. https://doi.org/10.1109/safeprocess45799.2019.9213341

Jung, D., & Frisk, E. (2018). Residual selection for fault detection and isolation using convex optimization. *Automatica*, *97*, 143–149. https://doi.org/10.1016/j.automatica.2018.08.006

Kim, W., & Katipamula, S. (2017). A review of fault detection and diagnostics methods for building systems. *Science and Technology for the Built Environment*, *24*(1), 3–21. https://doi.org/10.1080/23744731.2017.1318008

Llopis-Mengual, B., Yuill, D. P., & Navarro-Peris, E. (2025). Fault detection and diagnosis algorithm for multiple simultaneous faults in residential air-conditioning systems: Development, validation study and critical analysis. *Applied Thermal Engineering*, *269*, 125975. https://doi.org/10.1016/j.applthermaleng.2025.125975

Lu, S., Sian, H., Wang, M., & Kuo, C. (2021). Fault diagnosis of power capacitors using a convolutional neural network combined with the chaotic synchronisation method and the empirical mode decomposition method. *IET Science, Measurement & Technology*, *15*(7), 551–561. https://doi.org/10.1049/smt2.12056

Paolini, M., Savino Piscitelli, M., & Capozzoli, A. (2025). A label-free hybrid fault detection and diagnosis approach for HVAC systems using bayesian networks. *Energy and Buildings*, *351*, 116658. https://doi.org/10.1016/j.enbuild.2025.116658

Park, K.-S., Kim, J. H., & Park, C. S. (2026). Explainable fault detection and diagnosis in HVAC systems using a Koopman operator with normal data. *Energy and Buildings*, *350*, 116580. https://doi.org/10.1016/j.enbuild.2025.116580

Service Line Warranties of Canada. (2024, October 25). *State of the home: Survey shows three-quarters of homeowners face home repair emergencies*.

https://servicelinewarranties.ca/state-of-the-home-survey-shows-three-quarters-of-homeowners-face-home-repair-emergencies/

Statistics Canada. (2023, January 13). *The heat is on: How Canadians heat their home during the winter*. https://www.statcan.gc.ca/o1/en/plus/2717-heat-how-canadians-heat-their-home-during-winter

Wang, Z., Qin, Y., Kong, Y., Wang, L., Leng, Q., & Zhang, C. (2025). Advanced fault detection, diagnosis and prognosis in HVAC systems: Lifecycle insight, key challenges, and promising approaches. *Renewable and Sustainable Energy Reviews*, *219*, 115867. https://doi.org/10.1016/j.rser.2025.115867

Yan, Y., Sun, J., Yu, C., Sun, X., Qi Wu, E., Li, T., Cai, J., & David Cheok, A. (2024). An unknow fault diagnosis Scheme: A novel random deep forest for fault diagnosis of HVACs. *Energy and Buildings*, *321*, 114660. https://doi.org/10.1016/j.enbuild.2024.114660

Zhao, Y., Li, H., & Wen, J. (2019). A review of fault detection and diagnostics methods for HVAC systems. Energy and Buildings, 202, 109–131.

APPENDIX A: REQUIREMENTS

ID	MoSCoW Priority	Category	Requirement Description	Acceptance Criteria	Verification	Comments
Functional						
F1	Must Have	Data Ingestion	Ingest HVAC pump sensor data from .csv files (liquid line pressure, temperature, etc.).	Data is read and processed from a standard-format .csv file.	Run file read, check schema, sample rows.	Primary ingestion method.
F2	Must Have	Data Storage	Store historical .csv files in a local or shared directory.	Files are consistently named and retrievable.	Folder audit and read script.	Useful for batch training and inspection.
F3	Must Have	Fault Classifier (Supervised)	Use complete historical .csv cycles to classify fault presence (binary).	Model outputs ≥80% F1 on labeled validation set.	Model performance reports.	Trains on full HVAC pump lifecycle data.
F4	Must Have	Early Fault Prediction	Simulate streaming by processing .csv data in chunks to predict potential faults.	System flags potential faults partway through the cycle.	Compare early alerts to the final fault label.	Simulates live ingestion using sliding windows (Isolation Forest)
F5	Must Have	Feature Engineering	Create meaningful features from liquid line pressure, power, temperature data. Include time-series derivatives, rolling statistics, and buffer zones to capture subtle signal variation.	Feature vectors computed for both models.	Inspect feature set, correlations.	Shared across historical and real-time models. Also, move beyond simple aggregations to incorporate time-window patterns and variability that enhance fault detection.

F6	Must Have	Visualization Dashboard	Upload parquet file to display sensor signals, model results, and alerts.	Dashboard shows plots, classification outcome, and any warnings. Acceptable to AlertLabs and provides value.	File & model upload test and visual inspection.	Built using Streamlit partnered with models and HVAC data.
F7	Must Have	Fault Alert	Raise alert when fault risk crosses a threshold during ingestion simulation.	Alert is logged or displayed during the read.	Inject known faulty samples.	Display visually or via a UI log for testing.
F8	Must Have	Visualization Dashboard Content	Dashboard must allow for cell based evaluation and fleet level analysis.	Understandable single unit analysis and fleet.	User validation & Testing	Show curves and ML metrics for one cell and where this cell is faulting in time series data, likewise for fleet data.
F9	Must Have	Visualization Dashboard Configuratio n	Configuration of models being shown on dashboard must be possible for comparisons	Dashboard must load all used models efficiently without fault	No error codes or improper display for all models.	Displays same comparable data for all models so stakeholders can collect and use insights.
F10	Must Have	Alert Threshold Control	The dashboard must allow users to adjust the fault alert threshold to tune the precision vs. recall tradeoff.	Threshold control is available in the UI; changing it visibly updates alerts and results.	Adjust threshold and verify alert counts change accordingly.	Directly tied to the design constraint of prioritizing precision over recall.
F11	Must Have	Predicted Failure Onset Display	The dashboard must display the predicted time or cycle point at which a fault is expected to begin.	Predicted onset is shown alongside classification results for each unit.	Compare displayed onset to ground truth labels on validation data.	Provides actionable lead time for maintenance decisions.
Non-Functional						
NF1	Must Have	Usability	Users must be able to upload .csv files or run CLI jobs.	Drag-drop interface or terminal command works.	Test different file sizes and formats.	Simplifies testing and usage.
NF2	Must Have	Performance	The real-time prediction step must respond within ~20 seconds.	Each chunk processed within target time.	Time logging in Python.	Simulated ingestion should be responsive.
NF5	Must Have	Maintainabili ty	The codebase must be documented and modular to allow smooth transition to the company team.	README, code comments, function separation, docstrings.	Peer review or sponsor walkthrough.	Critical for handoff and long-term usability.
NF7	Must Have	Reproducibil	All models and results	Same results produced	Try rerunning	Essential for QA,

		ity	should be reproducible from scripts and config files.	with fixed seed, config inputs.	pipeline end-to-end.	validation, and academic/professional integrity.
NF10	Must Have	Precision Emphasis	Model outputs and default thresholds must be configured to favour precision over recall, minimizing false positives.	Default settings yield precision greater than recall on validation data.	Evaluate precision/recall on labeled test set with default config.	Reflects AlertLabs' operational preference to avoid nuisance alerts.
NF3	Should Have	Portability	Code should run on any machine with Python and basic packages.	No external dependencies outside pandas, scikit-learn, etc.	Test in a clean environment.	Supports collaboration and handoff.
NF6	Should Have	Scalability	The system should allow scaling to multiple HVAC streams or larger files.	Handles larger .csv files or parallel batch runs.	Stress test with larger datasets.	Enables future application at scale with more cells or hardware setups.
NF8	Should Have	Code Portability	The system should work across Windows/macOS/Linux with minimal setup.	Runs with one requirements.txt and instructions.	Test on multiple OS environments.	Helps for both deployment and cross-team collaboration.
NF11	Should Have	Deployment Transferability	The system must be packageable and deployable in a company environment with minimal setup beyond a requirements file and README.	A new team member can deploy the full system without external support.	Internal deployment test or sponsor walkthrough on a clean machine.	Complements NF5; focuses on operational deployment rather than just code handoff.
NF9	Could Have	Configurability	Model and ingestion parameters should be changeable without editing core code.	Parameters set via config file or CLI args.	Change parameters, verify output change.	Optional, it improves reusability if scope/time allows.
Model Requirements						
M1	Must Have	Supervised Model	Classify if HVAC pump is faulted from full .csv cycle.	Binary label	Accuracy, F1, Confusion Matrix	
M2	Must Have	Early Detection	Predict fault risk during ingestion (e.g., mid-cycle).	Binary flag or risk score	Precision at N, Lead Time, AUC	Simulate using chunked .csv parsing.
M3	Must Have	Shared Feature Set	Use time-series features (rolling stats, derivatives, slope, temperature trends, etc.) that capture signal variation over time	Feature vector	Feature importance scores	Includes time-window patterns and variability.

M8	Must Have	Unsupervised time-series model	Detects the period of time where fault is occurring.	Fault score threshold	F1, Accuracy	
M6	Should Have	Generalization	Models should learn patterns across varying test conditions and hardware.	Stable fault predictions across scenarios	Test accuracy across HVAC pump types	Add HVAC pump ID, normalized liquid line pressure as input features. Possibly use domain adaptation techniques.
M7	Should Have	Offline Filtering	Offline (inactive) HVAC unit windows should be excluded from training and prediction.	Filtered data	Accuracy without false offline alerts	Offline periods can be identified by flat values, or provided metadata.
M4	Could Have	Explainability	Identify top features driving fault predictions (e.g., SHAP, z-score).	List of contributing features	Visual attribution clarity	Useful for transparency in the dashboard.
M5	Could Have	Recommendation	Suggest next action post-fault (stop test, check temperature, etc.)	Text or rule	Manual validation	Stretch goal depending on bandwidth and expert input.
UI/UX Requirements						
UI1	Must Have	UI/UX	The dashboard must use distinct visual styling to clearly differentiate normal, warning, and fault states across all views.	Normal, warning, and fault states are visually unambiguous on all charts and indicators.	Visual inspection and user feedback during testing.	Yellow for warning, red for alert…
UI2	Should Have	UI/UX	All time-series and results charts must support interactive features including zoom, pan, hover tooltips, and filtering.	User can zoom into a time window, hover for exact values, and filter by unit or model.	Manual interaction testing across all chart types.	Streamlit via Plotly
UI3	Should Have	UI/UX	The dashboard must present a clean, organized layout with logical grouping of panels and minimal visual clutter.	Key metrics and charts are visible without excessive scrolling; stakeholder review confirms intuitiveness.	User walkthrough, PPC1, PPC2	
UI4	Should Have	UI/UX	The dashboard must display a confidence score or probability estimate alongside each fault classification	Confidence value is shown per prediction and updates with each new result.	Compare displayed confidence to raw model output values.	Helps operators gauge reliability before acting on an alert.

			result.			
UI5	Should Have	UI/UX	The dashboard must display key model performance metrics (precision, recall, F1, AUC) per model for stakeholder review.	Metrics panel is visible in the dashboard and values match offline evaluation reports.	Cross-check displayed values against model evaluation scripts.	Supports model comparison (F9) and transparency for industry stakeholders.
UI6	Should Have	UI/UX	All user-facing labels, tooltips, and descriptions must use plain language appropriate for non-technical operators.	Non-technical stakeholder review confirms terminology is accessible and clear.	Walkthrough with a non-technical sponsor or operator.	Model names, etc. Should be explained for context.
U17	Should Have	UI/UX	Terminology Clarification: The system must provide clear definitions for domain-specific terms (e.g., “core”, “fleet”)	Definitions for key domain-specific terms (e.g., “core”, “fleet”) are accessible within the interface (e.g., tooltips, labels, or info icons)	Stakeholder walkthrough confirms clarity	Reduces confusion for non-domain users.
U18	Should Have	UI/UX	Sensor-Model Alignment Visualization: Model outputs must be visually aligned with underlying sensor data.	Model predictions are displayed alongside corresponding sensor data in a synchronized timeline view	Stakeholder validation: engineers confirm usefulness for interpreting predictions	Enables validation of predictions and improves user trust.
U19	Should Have	UI/UX	Unit-Level Card-Based Visualization: Replace dense tables with card-based summaries showing risk, alerts, and trends.	Each unit is represented by a card displaying key metrics (e.g., fault probability, alert status, trend, recent alerts)	Usability testing task: PPC2	Improves scannability and reduces cognitive load.

APPENDIX B: REQUIREMENT CHANGE LOG

Date	**Change (High-Level)**	**Description (High-Level)**
Please refer to the following document, which includes the full change log history: https://docs.google.com/document/d/1Rsz6qzvcSxGbaBC5uOPvwo3IjwaaJaoGrHwodW_SzRs/edit?tab=t.0 The project initially was formulated with a different scope and has been modified since.		
February 5th, 2026	* Project Scope Change Removed irrelevant Functional Requirements and one Non-Functional Requirement	• Removed F8: Allow manual tagging of false positives/negatives for retraining. • Removed F9: Provide basic suggestions when a fault is detected (e.g., stop test, review voltage). • Removed F10: The system should support

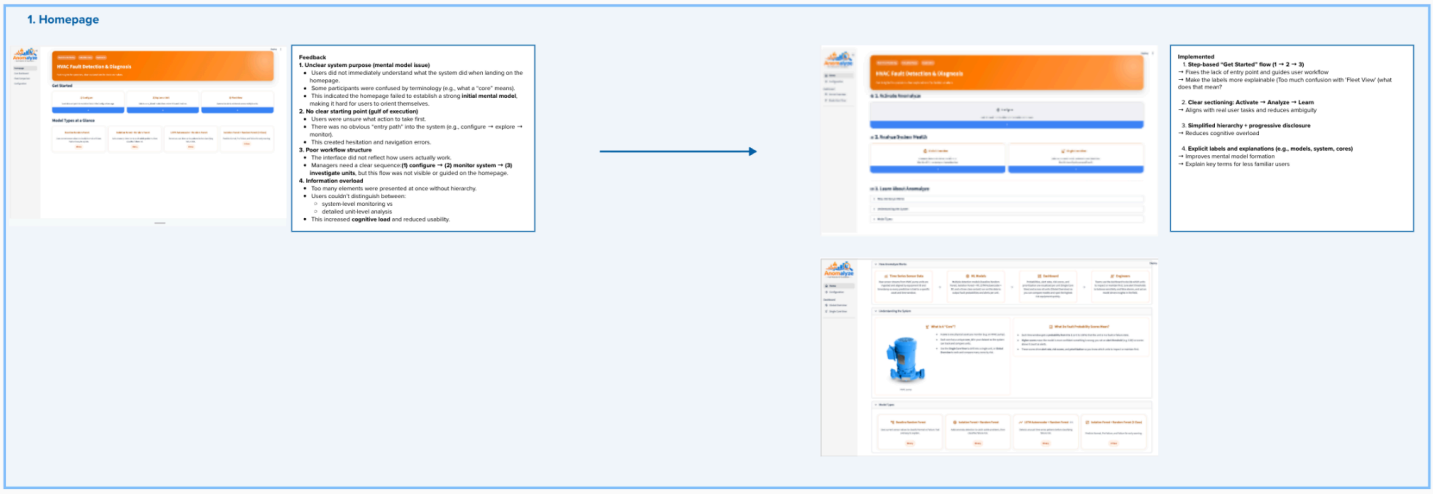
		batteries tested under different voltage/current conditions. ● Removed F11: The system should recognize and ignore known offline periods (e.g., daily maintenance windows). ● Removed NF4: In future, enable optional Azure ingestion or deployment.
February 5th, 2026	* Project Scope Change Added new Functional Requirements that are better aligned with Dashboard focused deliverables to AlertLabs.	● Added F8: Dashboard must allow for cell based evaluation and fleet level analysis. ● Added F9: Configuration of models being shown on the dashboard must be possible for comparisons. ● Added F10: The dashboard must allow users to adjust the fault alert threshold to tune the precision vs. recall tradeoff. ● Added F11: The dashboard must display the predicted time or cycle point at which a fault is expected to begin. ● Added NF10: Model outputs and default thresholds must be configured to favour precision over recall, minimizing false positives. ● Added NF11: The system must be packageable and deployable in a company environment with minimal setup beyond a requirements file and README. ● Added UI1: The dashboard must use distinct visual styling to clearly differentiate normal, warning, and fault states across all views. ● Added UI2: All time-series and results charts must support interactive features including zoom, pan, hover tooltips, and filtering. ● Added UI3: The dashboard must present a clean, organized layout with logical grouping of panels and minimal visual clutter. ● Added UI4: The dashboard must display a confidence score or probability estimate alongside each fault classification result. ● Added UI5: The dashboard must display key model performance metrics (precision, recall, F1, AUC) per model for stakeholder review. ● Added UI6: All user-facing labels, tooltips, and descriptions must use plain language appropriate for non-technical operators.
March 1st	User-Centred Design Iteration and Requirement Refinement Following think-aloud usability testing, expert advisory review, and peer feedback, several usability gaps were identified related to mental model formation, workflow clarity, interpretability, and information hierarchy. These findings prompted the refinement and expansion of UI/UX and functional requirements to better align	● UI1 (Updated): Visual State Differentiation - Enhanced to include consistent colour-coded prioritization for triage (e.g., green, yellow, red). ● UI3 (Updated): Information Hierarchy and Layout- Refined to enforce structured layering: Top: system summary, Middle: alerts and context, Bottom: unit-level diagnostics Justification: Reduces cognitive load and

	the system with real-world operational workflows and decision-support needs.	improves scannability. • UI4 (Updated): Model Output Interpretability - Expanded to include threshold lines, alert labels, and contextual explanations of risk scores. Justification: Addresses misinterpretation of probabilities. • UI7: Terminology Clarification: The system must provide clear definitions for domain-specific terms (e.g., "core", "fleet"). Justification: Reduces confusion for non-domain users. • UI18: Sensor-Model Alignment Visualization: Model outputs must be visually aligned with underlying sensor data. Justification: Enables validation of predictions and improves user trust. • UI19: Unit-Level Card-Based Visualization: Replace dense tables with card-based summaries showing risk, alerts, and trends. Justification: Improves scannability and reduces cognitive load.

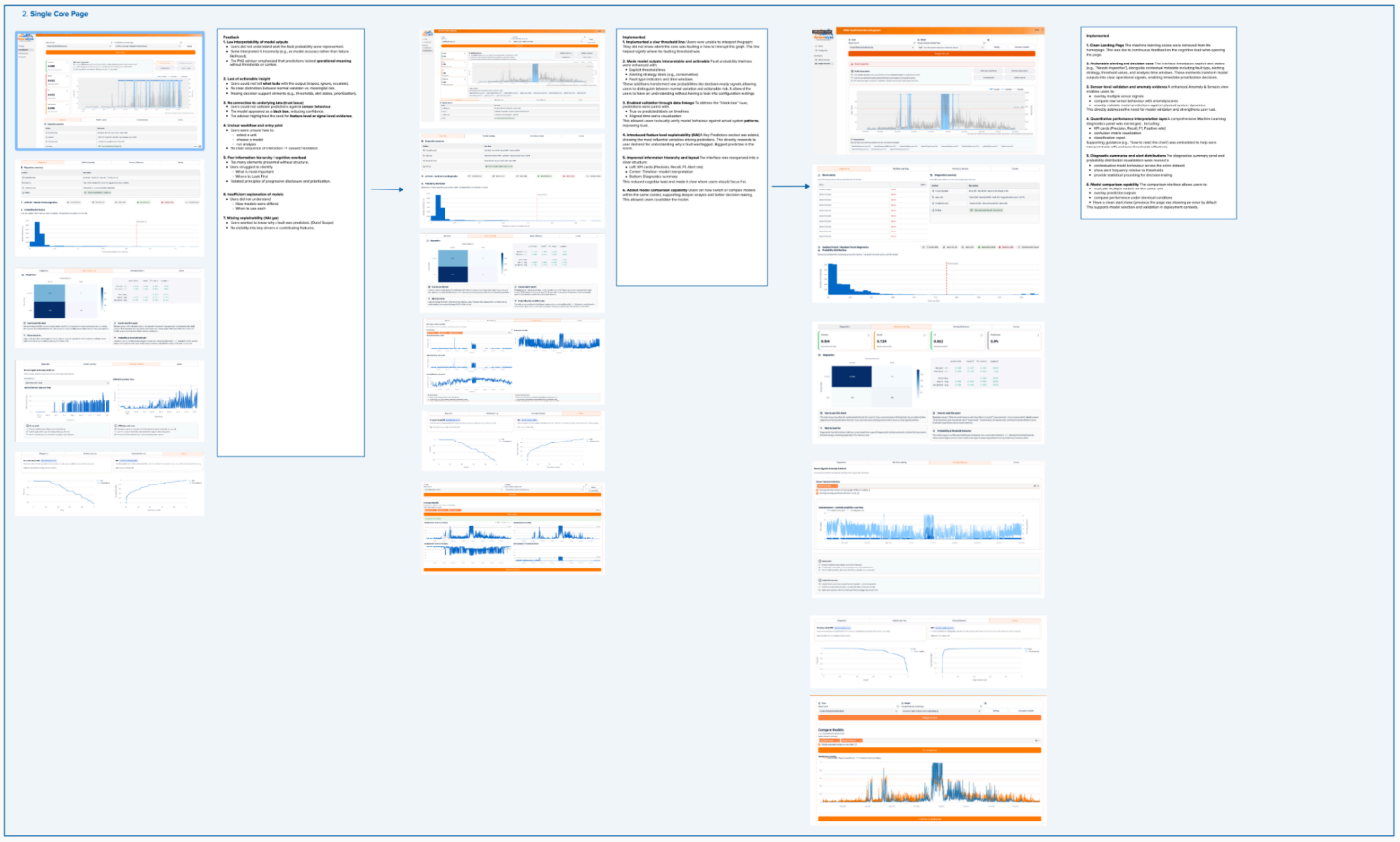
APPENDIX C: DESIGN ITERATION LOG

Full Scale Iterations Mural: https://app.mural.co/t/ella1832/m/ella1832/1733628653234/bb39d97779bff3af4927a1975d641e60f91cb5e7?sender=ubd088d7d36cb45d784625114

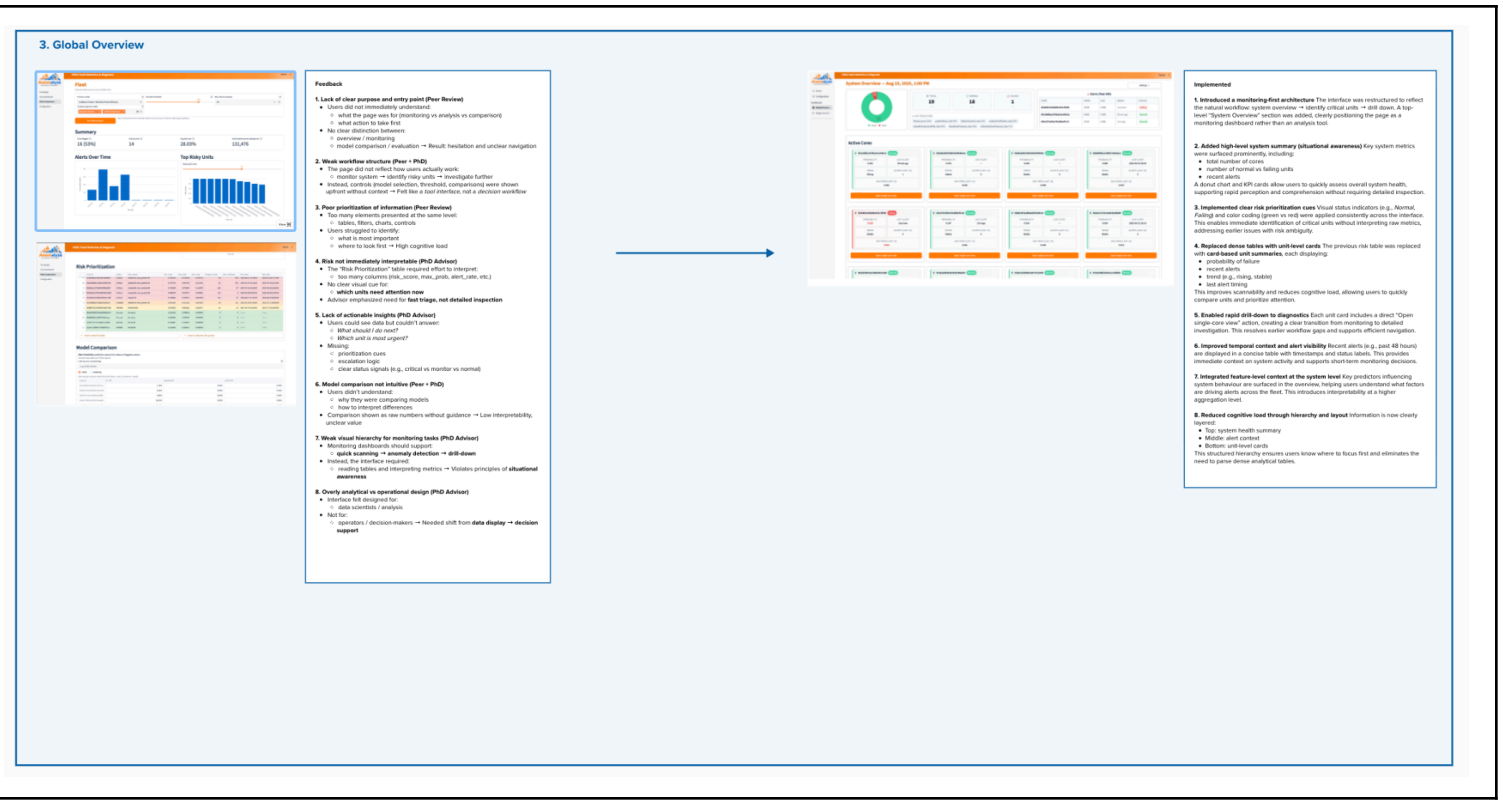
Homepage



Single Core Page



Global Overview Page



APPENDIX D: USABILITY STUDY QUESTIONS

Qualtrics Survey: https://docs.google.com/spreadsheets/d/11qy6nVcsANDRvTYKNFZ2UiWBUqu-idnJ/edit?usp=sharing&ouid=101712507643155375837&rtpof=true&sd=true

- When you first open the homepage, what do you think this system does?
- At any point did you feel unsure about what the next step in the workflow should be?
- When you saw the fault risk score, what did you think it meant?
- Did the dashboard clearly explain why the model raised an alert?
- Were the time-series charts easy to interpret? What, if anything, was confusing?
- In the global view comparison view, could you quickly identify the most critical units?

APPENDIX E: ADDITIONAL RESOURCES

Literature Review: Literature Review HVAC Faults

Video Demo: Anomalyze Dashboard

MSE 401 EOT Report (e-Zinc): Group 9 - EOT Design Review

Deployed Link (Private): https://hvac-pumps-fault-detection-alertlabs.streamlit.app/

- Please let us know if access to the deployed app is required. It will require company data to access. We have attached the zip file of the repository and & video demo.

APPENDIX F: MACHINE LEARNING MODELS

Model	Rationale
Random Forest (Balanced)	Used as a strong supervised baseline because Random Forest performs well on structured tabular data, handles nonlinear relationships, and remains relatively interpretable through feature importance. It was included to evaluate how well a standard supervised model could classify faulty vs. normal systems using engineered sensor features.
Random Forest (Recall-Focused)	Tested to examine whether shifting the model toward higher recall would improve early fault capture. This was relevant because missed faults are costly; however, increasing recall can also increase false positives, making this an important trade-off to evaluate.
Random Forest (Precision-Focused)	Tested to evaluate the opposite trade-off, where the model is optimized to minimize false alarms. This was operationally relevant because Alert Labs prefers to avoid nuisance alerts, but overly emphasizing precision can reduce sensitivity to true faults.
Isolation Forest	Considered because it is well suited to anomaly detection in settings with limited or coarse labels. It can identify unusual operating behaviour without requiring exact fault timestamps, making it appropriate for noisy HVAC time-series data where early degradation may not yet be labeled clearly.
LSTM Autoencoder	Tested because sequence-based models can capture temporal dependencies and reconstruct normal system behaviour, potentially making them useful for identifying evolving faults in time-series data. It was included to assess whether additional temporal modelling improved detection relative to simpler tabular approaches.
Isolation Forest + Random Forest (Hybrid)	Selected as the final model because it combines the strengths of unsupervised anomaly detection and supervised classification. Isolation Forest captures deviations from normal behaviour, while Random Forest uses these anomaly signals alongside engineered features to distinguish meaningful faults from benign variation. This made it better suited to the project's noisy data, gradual fault progression, and limited fault-time labels.

APPENDIX G: GENERATIVE AI DECLARATION

OpenAI. (2025). *ChatGPT* (GPT-5.3) [Large language model]. https://chat.openai.com/

Usage	**Explaination & Example**
Report Writing	Explanation: ● Used to guide the structuring and formulation of written content ● Assisted in improving conciseness to meet word count limit ● Helped refine explanations for clarity and further logical flow Example: https://chatgpt.com/share/69cfefba-7a74-832a-8db6-a5bf2f85e2f4 Example: https://chatgpt.com/share/69d2faf8-3f40-832e-8694-7bcdb1912446

UI Dashboard Coding	Explaination: ● AI was used to help code the dashboard. AI was prompted with specific problems or debugging. Prompts ● In the Anomlay & Sensors Section is there a way to overlay features? Like putting them on the same graph (Different colors?) ● By default I need you to load all the files. I don't want the user to have to configure anything" ● Can you make the borders the same color as the header bar? ● When I change models on the global page it shows the UI rendering. Like it is unformatted and looks really messy, how can that be stopped? ● That's much better. Small things. It shows the updating spinner. Then it goes away, it flashes then loads. Could that be made more smooth ● When I click Analyze this unit I see the need inspection block twice. Do not use the st.toast Example: https://chatgpt.com/share/69d2faf8-3f40-832e-8694-7bcdb1912446 Example: https://chatgpt.com/share/69d2fbc4-746c-8328-98a9-d0f78f56ea7b
Model Validation	Explaination: ● Used to help understand Model Outputs ● Used to help clean up repo structure & format ● Used to help brainstorm potential next steps Prompts: ● Can you please clean up the repo. Put everything into folders please. And make a clean README. ● Why would global scaling make more sense than local scaling ● Can I get the time scale on these? I'm curious to how zoomed in they are I maybe want to see like month snap shots ● In the Visualize Cell Can you make a really Pretty Graphic (Less Numbers) For the Core I provided. I need it to clearly show that liquidLinePressure_max is a validation technique for the model ● Could I get the same graph for powerfactor ● ok great this is good for me to validate on. Can you clean up the notebook? I want it to be clear. I want to test different types of scaling. StandardScaler, MinMaxScaler, RobustScaler and local vs global scaling for time and F1 score
System Understanding	Explanation: ● Used to help understand Battery & HVAC systems ● Used to develop simple example explanation of systems Prompts: ● Explain an HVAC pump to me like I'm 5 ● Explain to me what the liquid line is ● What are the best ways to explain this to a non technical audience ● What is the full process for heating and cooling using an HVAC unit, explain whereabouts failures tend to happen

Word Count: 4334