Mälardalen University
School of Innovation Design and Engineering
Eskilstuna, Sweden

Bachelor's thesis in Computer Science with a specialization in Intelligent Systems

ANOMALY DETECTION IN MANUFACTURING FOR QUALITY CONTROL

Fabian Nilsson
fabian.malkolm.nilsson@gmail.com

Dennis Kyrk
grownery.kyrk@gmail.com

Examiner: Kristian Sandström
Mälardalen University, Västerås, Sweden

Supervisor(s): Moris Behnam, John Moberg
Mälardalen University, Västerås and MITC, Eskilstuna

Company Supervisor(s): Gustav Persson
Alfa Laval, Lund

23/06/2025

Abstract

Modern heat exchanger factories test every unit by recording a one-dimensional spatial sequence and relying on a human operator to spot anomalies, a slow and repetitive process that can generate false alarms or allow faulty exchangers to reach customers. This thesis investigates whether supervised machine learning can automate inspections and improve reliability, addressing two core challenges: extreme class imbalance and variable sequence lengths. For tree-based models, we engineered 149 statistical, spectral and shape features. For deep learning, we resampled each signal to 500 points and trained with focal loss. We compared XGBoost, a convolutional neural network and Transformer hybrid called ConvTran, and four ensemble strategies under stratified five-fold cross-validation. Focusing on the minority (defect) class, XGBoost achieved an F1 score of 73 percent, precision of 73 percent, and a recall of 76 percent. ConvTran detected more faults (recall at 90 percent) but produced many false alarms (precision at 38 percent). By blending their strengths, the weighted adaptive ensemble reached precision of 82 percent, recall of 90 percent and an F1 score of 86 percent. A model-switching ensemble increased precision to 89 percent with recall at 84 percent, achieving overall accuracy of 98.6 percent. These findings demonstrate that automated anomaly detection is feasible with further refinement. A hybrid workflow that flags low-confidence predictions for manual inspection while fully automating high-confidence predictions appears most promising.

Contents

List of Figures

List of Tables

1. Introduction

This thesis explores if supervised machine learning models can automate, replace, or work in collaboration with manual quality control processes to improve heat exchanger manufacturing quality assurance. A leading heat exchanger manufacturing company currently uses a testing machine that performs measurements on each unit. These measurements are then inspected manually to determine whether a heat exchanger passes or fails. The manual classification of each unit is time-consuming and because of factors like long shifts, varying skill levels and human errors the consistency and accuracy may vary. Therefore, this work serves as a proof of concept to investigate how machine learning approaches could enhance the existing quality control framework. Anomaly detection is the task of identifying observations that depart from the regular behaviour of a system. So given a dataset containing mostly normal examples, anomaly detection algorithms seek to identify rare instances that differ from the majority. The field can be separated into supervised, semi-supervised and unsupervised settings [1]. Anomaly detection in heat exchanger production is critical for identifying faulty units early and preventing them from reaching customers. The dataset that the company has provided us with consists of three types of data.

1. Features describing the physical properties of the heat exchangers. They can be seen as static features that are either categorical or scalar. In this report, we will be referring to these static features as DUT(Device under test)[1] features. The most important DUT features are DUT 6 and DUT 7. DUT 6 is the most important categorical DUT feature and reflect a wide range of different heat exchanger types that have significantly different physical properties. DUT 7 is the most important scalar DUT feature and roughly reflect the size of the heat exchangers.

2. The settings of the testing machine that affect the characteristics of the sequence, such as sampling rate.

3. The measurement data captured by the testing machine at a constant spatial sampling rate.

Earlier studies show that convolutional networks such as ResNet and InceptionTime ranked as top 4 to reach state-of-the-art accuracy in anomaly detection across 22 manufacturing datasets n [2]. On the other hand, XGBoost outperforms most machine learning models in both training time and accuracy when tested on multiple large classification datasets that are not necessarily related to anomaly detection [3]. Two common limitations are often concluded in similar studies. First, many supervised machine learning models collapse to predict the majority class when minority class proportion falls below 10 percent [4], [5]. Second, as of our knowledge, most publications don't discuss wether the manufactured units have different number of samples per sequence. In this paper, we address both limitations. The goal is to develop and evaluate a model that automatically classifies heat exchangers based on univariate spatial sequence measurements. A key challenge is that these univariate spatial sequences vary in length depending on the physical properties of each heat exchanger and testing machine settings.

The research questions that will be addressed in this study are:

RQ1: How well can XGBoost and ConvTran separately identify defective heat exchangers in a highly imbalanced industrial univariate spatial dataset?

RQ2: How well can an ensemble that combines a deep network (ConvTran) with a tree-based tabular classifier (XGBoost) identify defective heat exchangers in a highly imbalanced industrial univariate spatial dataset?

This thesis seeks to deliver the following:

1. a reproducible quality-control pipeline for variable-length sequences,

[1]This thesis was conducted under an NDA contract; detailed information about the dataset is anonymized and confidential.

2. a quantitative comparison between XGBoost, ConvTran and their ensemble on the companies measurement data, and

3. a way of handling class imbalance and varying sequence length.

The intended audience is engineering and computer science students and practitioners who have basic knowledge of machine learning but limited familiarity with anomaly detection in industrial processing data.

1. **Background** introduces the reader to the study's context. It explains key technical terms, describes the general problem setting, and clarifies why the topic matters for industrial quality control. This section prepares the ground for everything that follows.

2. **Related Work** places the project within the wider research landscape. By summarising earlier studies and pointing out open questions, it shows how the present work extends the field and motivates the chosen approach.

3. **Method** sets out the overall study design. It explains how data are handled, how models are framed, and which evaluation criteria guide the experiments, giving a clear blueprint for the investigation.

4. **Implementation** turns the method into practice. It records practical decisions such as code structure, software libraries, hardware, and parameter settings, so that the experimental pipeline can be reproduced.

5. **Results** reports the quantitative findings without interpretation. It presents the core performance metrics and any supporting figures in a structured manner, offering a transparent view of what the models achieved.

6. **Discussion** interprets those findings in relation to the research questions. It reflects on practical implications, examines limitations, and suggests directions for further study, linking the results back to the broader field.

7. **Conclusion** summarises the main contributions of the thesis. It answers the research questions concisely and indicates how the proposed approach can support reliable inspection of units in an industrial setting.

2. Background

2.1 Dataset Characteristics

The dataset consists of measurements from a quality control testing machine. The measurements form a univariate spatial sequence, which in this case means that data is represented as a series of measurements over a distance. So, for each heat exchanger, a sequence of measurement $S = \{S_1, S_2, \ldots, S_n\}$ is collected where each $S_{(i)}$ represents a measurement at a constant sampling rate along a spatial path. Both the DUT1 features of the heat exchangers and the settings of the test machine have a strong influence on the characteristics of the sequences. The number of measurements varies from 70 to 1200 measurements depending on the properties of the heat exchangers as well as the settings of the test machine. These factors also affect how well different classification models can detect defects, making it important to account for them when designing and evaluating anomaly detection methods. The main challenge is to reliably identify deviations in these sequences. We will refer to each heat exchanger in the dataset as a unit, which is classified as either normal (majority class) or defect (minority class). Each measured profile is a sequence of univariate spatial data, and all anomaly detection and subsequent analysis operate on these sequences.

As shown in Figure 1,there are two types of defects that can occur in every sequence. Defect type one manifests as a sudden, sharp peak in the signal, characterized by a rapid increase in amplitude followed almost immediately by an equally rapid decrease. This creates an isolated spike whose position varies randomly along the measurement, resembling a brief but pronounced disturbance. This type of defect could represent sudden disturbances or momentary failures in a manufacturing process. Defect type two presents as a prolonged depression in the signal, often occuring in the latter half of the measurement. The valley is characterized by a gradual descent followed by a low period and then a recovery. After the valley, the signal recovers and returns to normal levels. Defect type two could represent a sustained deviation in process parameters or a temporary but significant shift in operating conditions.

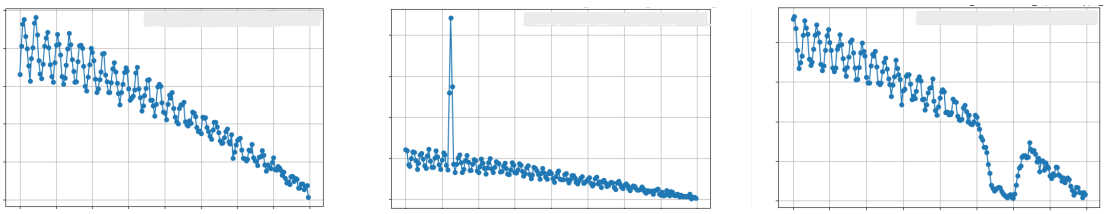

Figure 1: Synthetic sequence from a normal unit (left), Defect type 1 (middle), and Defect type 2 (right)

Industrial datasets are usually imbalanced, with far fewer defective units than non-defective ones[6]. Our dataset is no different; the dataset's minority class consists of about 5 percent of the entire dataset.

2.2 Evaluation Metrics

Standard accuracy is therefore misleading. Precision and recall for the minority class give clearer insight[4], and for that reason, our evaluation focuses mainly on minority precision and minority recall.

$$\text{Recall}_{\text{minority}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1}$$

$$\text{Precision}_{\text{minority}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{F1}_{\text{minority}} = 2 \times \frac{\text{Precision}_{\text{minority}} \times \text{Recall}_{\text{minority}}}{\text{Precision}_{\text{minority}} + \text{Recall}_{\text{minority}}} \tag{3}$$

Where TP represents true positives (defects correctly identified), FP represents false positives (normal units incorrectly classified as defects), and FN represents false negatives (defects mislabeled as normal).

2.3 Handling Class Imbalance

Class imbalance is a significant challenge in most machine learning applications where one class is significantly underrepresented in a dataset compared to the other class or classes[4]. This problem is very normal in manufacturing quality control contexts, where defective parts may only represent a small percentage of all manufactured units[7]. When working with imbalanced datasets, standard machine learning algorithms tend to bias towards the majority class by default. This can lead to bad performance where a model predict all data as normal which in our case would give it 95% accuracy which is generally considered very good for balanced datasets[8]. In manufacturing anomaly detection, this kind of bias is problematic since failing to identify defective parts typically carries much higher costs than misclassifying normal parts as defective[9].

2.3.1 Resampling Techniques

Resampling techniques adjust the class distribution in the training set to reduce the imbalance between classes[10]. Undersampling reduces the number of majority class instances. In our implementation, we used random undersampling to achieve a balanced 1:1 ratio between normal and defective samples. While simple and effective, this approach discards potentially valuable information from the normal samples. Oversampling increases the number of minority class instances. Simple duplication of minority samples can lead to overfitting, so more sophisticated techniques like SMOTE (Synthetic Minority Over-sampling Technique) are often used. SMOTE generates synthetic examples by interpolating between existing minority class samples in the feature space[11]. While effective for addressing class imbalance, SMOTE has some drawbacks. For example, it may generate unrealistic synthetic data, particularly in regions where classes overlap, which can lead to overfitting if the synthetic samples introduce noise into the training data.

2.3.2 Focal Loss

Focal loss is a modification of the standard cross-entropy loss that adds a factor to reduce the weight of well-classified examples and focus learning on harder, misclassified cases [12].

$$FL(p_t) = -\left(1 - p_t\right)^{\gamma} \log(p_t) \tag{4}$$

Equation (4) defines focal loss, where p_t is the model's predicted probability for the true class and γ is a non-negative focusing parameter. When $\gamma = 0$ focal loss becomes ordinary cross-entropy. As γ increases, well-classified examples with high p_t contribute less to the loss, and misclassified or hard examples with low p_t contribute more. In imbalanced industrial datasets most units are normal and the model learns to classify normal units. By applying focal loss with $\gamma > 0$ the many easy normal units have reduced impact on the gradient, while the fewer defect examples that the model still mislabels carry larger gradients. This shifts learning toward reducing false negatives and can improve recall on the minority defect class without resampling or complex cost-sensitive schemes. When training the model with focal loss, it focuses on subtle anomaly patterns that would otherwise be overshadowed by the majority class.

2.4 Handling Variable-Length Univariate Spatial Data

A challenge in our data is that each unit's univariate spatial data have variable length. The number of spatial measurements varies between units due to differences in models and testing parameters. This variable is a challenge because neural networks usually require fixed-length inputs to process

the input data[13].The reason for that is that the layers of neural networks are defined with fixed-size input weights with uniformly shaped tensors.

2.4.1 Linear Interpolation

Linear interpolation is a common method to convert variable-length univariate spatial data into fixed-length inputs for neural network models. Each unit's spatial measurements are mapped onto a grid of N equally spaced points, keeping the overall shape of the signal while standardising sequence length across different unit models and testing machine parameters. The TSinterpolator transformer in the sktime library fits scipy.interpolate.interp1d to each sequence and samples a user-defined number of points to enforce fixed length inputs[14].

2.4.2 Feature Engineering

Feature engineering can convert variable-length univariate spatial data into fixed-length number of features that can be used with machine learning models. We transformed the variable-length sequences into a fixed number of features using the TSFresh library [15] and custom features.

Automated extraction with the TSFresh library computes statistical summaries (mean, variance, skewness), frequency-domain metrics (FFT coefficients), and complexity measures (entropy, autocorrelation) for each sequence regardless of its original length. This approach extracts meaningful statistical properties from each sequence, creating consistent feature length data for classification. The result is a uniform feature matrix in which each unit is represented by the same number of features, enabling XGBoost to learn decision-tree ensembles without additional sequence-length handling.

2.5 Feature Engineering for Univariate Spatial Data

Feature engineering is the process of transforming raw data into meaningful features that can represent the underlying problem to predictive models [16]. While modern deep learning models such as Convolutional Neural Networks (CNNs), Long Short-term Memory Networks (LSTM), and transformers can effectively process raw univariate spatial data directly, traditional machine learning algorithms like decision trees, random forest and gradient boosting methods typically perform better with engineered features that capture relevant patterns. In contrast to deep learning approaches that can automatically extract time features and patterns from complex data, traditional approaches rely on domain-specific knowledge or meaningful feature engineering [17].

For heat exchanger univariate spatial data, feature engineering provides advantages compared to the raw data. In our case, the main benefit is interpretability: engineered features are generally more interpretable than raw data, making the model's decisions more transparent. Tree-based models like XGBoost can provide feature importance scores that show how each feature contributes to the prediction, offering clearer understanding to why a certain classification was made. This interpretability is especially valuable in manufacturing settings where understanding why a heat exchanger is classified as defective can be as or even more important than the detection itself. This is in contrast to more complex models like neural networks that can in many cases be considered a "black box" where the internal workings are complex and difficult to interpret [18].

2.6 Gradient Boosting and XGBoost

Gradient Boosting is an ensemble machine learning technique that builds a prediction model by combining multiple weak learning, typically decision trees[19]. The key principle is that each new tree corrects the error made by the previous ensemble of trees, gradually improving the model's accuracy.
XGBoost (eXtreme Gradient Boosting) [20] is an optimized implementation of the gradient boosting framework [19] that has become one of the most successful algorithms for structured/tabular data.

XGBoost offers several features that make it particularly effective for our dataset. One of its key strengths is the use of both L1 (Lasso) [21] and L2 (Ridge) [22] regularization, which helps prevent overfitting and improves the models ability to generalize to unseen data. Another advantage is

XGBoost's ability to perform well on imbalanced datasets, a common characteristic in anomaly detection tasks where defective units are relatively rare. Finally, XGBoost is computationally efficient, with fast training speeds and low memory requirements, making it well-suited for rapid experimentation and model iterations.

2.7 Stratified k-fold Cross-Validation

Stratified k-fold cross-validation is a variant of k-fold cross-validation that maintains the original class distribution across all folds. Unlike standard k-fold cross-validation, which randomly partitions data, stratified k-fold ensures that each fold contains approximately the same proportion of samples from each class as in the original dataset. This approach is important for imbalanced datasets, where random splitting could result in folds with very few or no samples from minority classes, which could lead to unreliable performance estimates during model validation.[23].

For example, with stratified 5-fold cross-validation, you first split the entire dataset into 5 splits (S_1, S_2, S_3, S_4, S_5), each having the same class distribution. Then you train the model 5 times, where each split serves as the test set once while the remaining four splits are used for training. The final performance estimate is calculated as the average across all five test results.

- Train: $\{S_2, S_3, S_4, S_5\}$, Test: $\{S_1\}$
- Train: $\{S_1, S_3, S_4, S_5\}$, Test: $\{S_2\}$
- Train: $\{S_1, S_2, S_4, S_5\}$, Test: $\{S_3\}$
- Train: $\{S_1, S_2, S_3, S_5\}$, Test: $\{S_4\}$
- Train: $\{S_1, S_2, S_3, S_4\}$, Test: $\{S_5\}$

2.8 Ensemble Models

Ensemble models combine the probability outputs of multiple classifiers to improve recall and precision for Defect units (minority class) and normal units (majority class). By merging complementary strengths of individual models, ensembles can better address class imbalance and variability in variable-length univariate spatial data [24].

2.8.1 Weighted averaging

Weighted averaging assigns each model a weight based on its performance on validation data. The ensemble probability is the sum of each model's probability multiplied by its weight, normalized to one. This method gives greater influence to models with higher recall or precision[25].

2.8.2 Rule-based combinations

Rule-based combinations apply predefined decision rules to model probabilities. For example, if one model's probability for a defect unit exceeds a threshold, its output may override others. These rules can be tailored to parameters[25].

2.9 Transformers

Transformers are a neural network architecture that uses self-attention to model relationships between all elements in an input sequence in parallel [26]. They replace recurrence with a mechanism that computes attention scores between query, key and value vectors, allowing the model to capture long-range dependencies more efficiently.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \tag{5}$$

Equation 5 computes self-attention by taking the dot product of the query matrix Q and the key

matrix K, dividing by the square root of the key dimension $\sqrt{d_k}$, applying softmax to get the attention weights, and then multiplying by the value matrix V.

According to various studies, transformer architectures have improved sequence modeling, showing strong performance in different tasks and settings such as natural language processing, image recognition, and time series analysis. [26]–[28]. Transformers primarily succeed due to their attention mechanism, which captures long-range dependencies. This contrasts with convolutional neural networks (CNNs) or recurrent neural networks (RNNs), which typically struggle with capturing global relationships.[26].

3. Related Work

3.1 ConvTran

Recently, integrating convolutional layers with transformer architectures has led to ConvTran, a hybrid model combining CNNs' capability to recognize local patterns and transformers' ability to understand global contexts [29]. Foumani et al. (2023) introduced ConvTran specifically for time series classification, proposing two new methods: time Absolute Position Encoding (tAPE) and efficient Relative Position Encoding (eRPE).

$$\tilde{\omega}_k \;=\; 10000^{-\frac{2k}{d_{\text{model}}}} \times \frac{d_{\text{model}}}{L} \tag{6}$$

$$p_i(2k) = \sin\big(i\,\tilde{\omega}_k\big), \quad p_i(2k+1) = \cos\big(i\,\tilde{\omega}_k\big). \tag{7}$$

Equation (6) shows the adjusted frequency term, scaling the standard sinusoidal frequency by the ratio of embedding dimension d_{model} to sequence length L. Equation (7) defines the absolute position embeddings using sine and cosine functions evaluated at each input position i.

$$A_{i,j} = \frac{\exp(e_{i,j})}{\sum_{m=1}^{L}\exp(e_{i,m})}, \quad e_{i,j} = \frac{(x_i W^Q)(x_j W^K)^\top}{\sqrt{d_z}} \tag{8}$$

$$z_i = \sum_{j=1}^{L}\big(A_{i,j} + w_{i-j}\big)\,x_j \tag{9}$$

In Equation (8), $A_{i,j}$ and $e_{i,j}$ represent the standard self-attention weight and its scaled dot-product score, respectively. Equation (9) then adds the learnable relative position scalar w_{i-j} to each attention weight before combining with the value vectors x_j.

The advantage of ConvTran for anomaly detection in industrial quality control applications lies in its ability to capture both short-range patterns and long-range dependencies within variable-length univariate sequences. This ability is important for detecting anomalies in manufacturing processes. The tAPE method improves absolute position encoding by considering both the series length and input embedding dimension, solves issues in traditional sinusoidal encodings. Also, eRPE provides a computationally efficient method for relative position encoding, which helps generalization and reduces overfitting. This is beneficial for datasets with imbalance between normal and anomalous classes[29]. The same paper compared ConvTran to earlier methods such as CNN-based models or traditional transformer-based models, and it scored higher precision and recall in anomaly detection tasks. It outperforms these models across multiple benchmark datasets. It's architecture utilizes CNNs for detecting local patterns while using the transformer's self-attention mechanism to handle complex global relationships within time series data. This combination improves performance in identifying manufacturing anomalies. Given the specific needs of anomaly detection in our heat exchanger dataset, such as managing highly imbalanced datasets, detecting subtle deviations, and generalizing across different heat exchanger models and physical dimensions, ConvTran stands out as a suitable state-of-the-art solution. Its demonstrated evaluation metrics on related time series classification tasks further support its use with our univariate distance series data.

3.2 Comparing Tree-based and Transformer Models

Boosted decision trees such as XGBoost remain a common starting point for industrial quality data because they handle tabular feature spaces and class imbalance with little preprocessing [20]. Recent comparisons suggest that transformer variants can match, or sometimes exceed, this baseline when tested on similar conditions as our variable-length univariate spatial data. Wachowiak et al. evaluated ConvTran, Random Forest and XGBoost on a multimodal, imbalanced time-series

dataset (14 percent minority, 86 percent majority). ConvTran reached 83.81 % cross-validation accuracy, while XGBoost peaked at 81.84 %, a two-point gap that held across hyper-parameter grids [30]. Even if the study is not in manufacturing, the similar class imbalance and direct ConvTran–XGBoost comparison make the findings relevant to our research questions.

Turning to manufacturing data, Deng et al.[31] applied a one-dimensional Vision Transformer hybridized with Random Forest (RF-1DViT) to continuous casting slab sensor signals, using entire steelmaking, refining, and casting process histories sliced into sequential univariate segments each labeled for non-metallic inclusion presence and position. Their RF-1DViT model attained an F1 score of 89.91%, outperforming logistic regression, k-nearest neighbors, support vector machines, random forest, AdaBoost, gradient boosting, multilayer perceptron, and standalone 1DViT baselines, while offering interpretability through feature-importance histograms. Additionally, t-SNE embeddings of the model's internal representations facilitated accurate defect localization within the slab slices. Together these studies indicate that transformer models are interesting to compare with XGBoost when classifying Defect and normal units in our heat-exchanger dataset.

3.3 Probability-Level Ensembles of Heterogeneous Models

When classifiers output calibrated probabilities, an ensemble can merge those scores to lift recall and precision for the minority class. Weighted averaging assigns higher influence to the model with better validation performance; rule-based schemes override averages under predefined conditions; region-based variants learn separate weights for subsets defined by static testing-machine parameters [24].

Haque and Soliman show the benefit of such fusion in sensor networks: a Transformer autoencoder detects candidate anomalies, Isolation Forest validates them, and XGBoost refines the class label [32]. The three-stage ensemble reaches 95 percent overall accuracy and cuts the false-negative rate for intrusions below 6 percent. Large-scale benchmarks that mix tree learners with deep models report similar gains, especially under severe class imbalance [33]. These findings motivate an ensemble that combines ConvTran and XGBoost probabilities to boost minority class recall (equation 1) without sacrificing majority class precision (equation 2) for our anomaly detection problem.

4. Method

Our work followed a sequence of stages that together formed the scientific method of the study. We began with a focused literature review on anomaly detection in manufacturing, which helped us identify model families that can handle univariate spatial data and the severe class imbalance in our dataset. Building on these insights, we explored the dataset in detail: descriptive statistics, symbolic regression, and correlation analyses of static testing-machine parameters guided the preprocessing, feature engineering, and model picking. With a clearer view of the data, we compared supervised and unsupervised approaches, considering the variable-length sequences and class imbalance. Baseline versions of each candidate were trained, and models that failed to improve minority class F1 were discarded. The remaining models were manually fine-tuned through stratified group K-fold cross-validation: this step made sure that performance remained stable across folds, seeds, and DUT-specific subgroups that reflect physical dimensions of the units. Evaluation focused on minority class precision, recall, and F1 scores, measured both globally and within specific DUT categories. These metrics were chosen because they best reflect the industrial trade-off between missing defects and incorrectly rejecting normal units. Finally, we combined the prediction probabilities of the top ConvTran and XGBoost models in an ensemble approach. This workflow was crafted to answer our research questions and to develop a proof of concept that is acceptable for automating the quality control in the heat exchanger production.

4.1 Data Preprocessing and Feature Extraction Implementation

Before any modeling could begin, the raw univariate spatial data were plotted and analyzed for reliability. Units whose labels were missing, corrupted, or known to originate from testing machine parameter settings used only during calibration were removed. This reduced the risk of training on sequences that do not reflect normal production conditions. Visual exploration helped us classify recurring defect patterns and guided later feature design.

4.1.1 Neural Network Preprocessing

All neural-network models in the study need inputs of equal length, yet the variable-length univariate spatial sequences collected from the testing machine range from only 70 up to 1 200 samples. We evaluated four approaches to standardisation: filling missing positions with zeros, padding with small random noise at both ends, padding with noise only at the tail, and rescaling every sequence to the global maximum length of 1 200 samples[34]. Making every unit this long placed a heavy burden on memory and training time and, more importantly, diluted information in short sequences. A signal originally containing 70 real measurements became a vector dominated by 1 130 zeros or synthetic noise, which reduced the model's ability to detect defects. The alternative that worked best was to resample every sequence to 500 samples by linear interpolation. This step was implemented with the interpolator transformer in the sktime library, which relies on SciPy's interpolate.interp1d tool[14]. Interpolation compresses longer sequences and stretches shorter ones while keeping the overall shape intact, and it limits the length of each input so that training remains tractable. Tests with target lengths between 200 and 1 000 samples produced very similar validation curves, so 500 samples were adopted as a balanced choice for every fold and every seed, ensuring fair and efficient comparisons across models.

4.1.2 Feature Engineering Pipeline

We implemented a progressive feature extraction approach applied directly to the raw variable-length univariate spatial data:

- Initial proof of concept using TSFresh automated feature extraction to establish that XGBoost could effectively work with feature-based approaches, yielding over 1000 initial features.
- Transition to custom domain-specific features (149 total) used specifically choosen for this dataset after multiple iterations.

We chose to move from automated TSFresh features to custom features because this provided better control over the feature engineering process and deeper understanding of which signal characteristics were most relevant for defect detection. The custom features also offered improved interpretability for industrial applications.

4.1.3 Class Imbalance Strategy Selection

For the class imbalance challange we implemented and compared two resampling approaches:

- Random undersampling to achieve 1:1 class ratio, creating approximately 3,000 training samples
- SMOTE oversampling with k=5 neighbors, generating approximately 60,000 training samples

Both approaches showed similar average performance across all cross-validation splits. While random undersampling provided computational advantages through drastically reduced training time by effectively creating a smaller dataset to train on (3000), we selected SMOTE as our final approach. Given that XGBoost training is not computationally expensive, we prioritized the stability and consistency of synthetic sample generation over computational efficiency. We wanted to avoid the risk of obtaining poor representative subsets due to the inherent randomness in undersampling, making SMOTE the safer choice with minimal additional computational cost when using XGBoost

4.2 Stratified Group K-Fold Cross-Validation

In order to evaluate our models' ability to detect anomalies in univariate spatial data from the testing machine, we adopted a stratified group K-fold cross-validation procedure. This method was chosen because it reflects the true class imbalance when estimating performance. Another reason is that it ensures that the entire distribution of the most important categorical DUT 6[1] feature is represented proportionally in every fold. For both ConvTran and XGboost, our two pipelines used scikit-learn's StratifiedGroupKFold[35] with a fixed random seed. Sequences were defined by unit identifiers, and stratification keys combined the defect/normal label with the DUT 6 category. At each iteration, one fold of units served as the test set, while the remaining folds formed a training pool. By aligning the number of folds, the random seed, and the composite stratification key across both scripts, we guarantee that each fold's train/test split of sequences is identical for ConvTran and XGBoost. This cross-validation approach enables a fair comparison of ConvTran and XGBoost performance on variable-length spatial sequences. It directly addresses our research questions about model robustness under class imbalance while ensuring reproducible results.

4.3 Hyperparameter Optimisation

The ConvTran hyperparameters were tuned through a manual, theory-driven process rather than an automated search. We began with values reported in the original ConvTran open source code [36] and adjusted them based on transformer design principles as well as the needs of our imbalanced data. At each step we trained the model across the stratified group K-fold splits and monitored validation metrics. Single hyperparameter settings that consistently improved the minority class F1 score across folds and random seeds were kept while others were rolled back. Training was limited to a point where validation loss no longer improved, which avoided over-fitting while keeping training time reasonable. This iterative method allowed us to incorporate theoretical insight, respond to empirical feedback, and arrive at a configuration that met our performance goals on univariate spatial data without an exhaustive search.

4.4 XGboost confidence threshold approach

We used a threshold approach where the XGBoost model would only classify very confident predictions where the output prediction probability was close to 0 for normal units and close to 1 for defect units. If the prediction probability was not within those extreme intervals, the sequences would remain unclassified. This approach was implemented to maximize performance

while allowing uncertain predictions to be flagged for human review. We analyzed the minority class precision-recall-coverage trade-off curves across different threshold values to visualize how increasing confidence thresholds affected model performance.

4.5 Ensemble Strategy Development

The ensemble combines the prediction probabilities from ConvTran and XGBoost to improve minority-class performance over the two stand alone models. Each unit was assigned prediction probability from both models, obtained with the best fold from the five-fold cross-validation. Several fusion ideas were prototyped: averaging, weighted averaging, confidence-based model switching, and rule sets that divide the joint probability space into regions.

5. Implementation

5.1 Unified Cross-Validation and Evaluation Pipeline

Both ConvTran and XGBoost are evaluated under the same data-handling routine to make their results directly comparable. The routine is managed by a small orchestration script that runs each model through an identical K-fold cycle while using a single master random seed.

1. The script loops over K folds. For every fold it creates separate output folders for the two models, passes the current fold index and the total number of folds as arguments, and records the seed so the partitions can be reproduced.

2. Inside each model the entire dataset is loaded and divided with StratifiedGroupKFold. The stratification key joins the defect/normal label with the categorical feature DUT 6[1]. All samples that belong to the same unit stay together, and every fold preserves the minority/majority class ratio as well as the distribution of DUT 6. The train portion of the fold is then split once more into training and validation sets with a second stratified split.

3. Each model is fitted only on the training split for the current fold and monitored on the fold's validation split. After training, performance is measured on the unseen test split of that fold.

4. For every fold the following numbers are stored:

   (a) overall accuracy,

   (b) confusion matrix,

   (c) precision recall, and F1-score for the minority class.

   Results are written to disk in structured files which enables averages and standard deviations to be calculated across folds. Metrics are recalculated for pre-defined subsets, such as individual values of DUT 6 or binned ranges of DUT 7. These metrics reveal how robust each model is under different operating conditions.

The common pipeline applies the same training, validation, and testing logic to both models. By determining the splits, the random seed, and the set of reported metrics it allows a fair, fold-by-fold comparison of ConvTran and XGBoost on the univariate spatial data.

5.2 XGBoost Pipeline Implementation

This pipeline trains an XGBoost classifier that separates defect units from normal units using the univariate spatial data captured by the testing machine. The script is started from the command line and all key paths, fold indices, and hyper-parameters are supplied as arguments, which makes every run reproducible. The XGBoost model was trained and evaluated using a 5-fold cross-validation strategy, similar to the approach used for the ConvTran model. Data was split into five folds with stratification to preserve class proportions and important DUT data across all splits. For each fold:

1. Feature extraction and scaling were performed only on the training set

2. To adress class imbalance we either used undersampling or SMOTE to balance the training data.

3. The model was trained on the balanced training set

4. Evaluation was performed on the untransformed validation and test sets

5.2.1 XGBoost hyperparameter tuning

Only minimal hyperparameter tuning was performed, as the model showed strong performance with near-default settings. The main parameters adjusted were learning rate, maximum tree depth, minimum child weight and subsample ratio.

After testing various configurations and performing multiple 5-fold cross-validation runs, the following set of parameters was found to work well:

1. **Number of Trees**: The model was built using 1100 individual decision trees to improve accuracy through boosting.
2. **Learning Rate**: A small step size was used when updating the model to ensure gradual learning and reduce the risk of overfitting.
3. **Maximum Tree Depth**: Each tree was limited to a maximum depth of 4 levels, controlling complexity and helping generalization.
4. **Minimum Child Weight**: A leaf node needed to have a minimum combined weight of 2 to be split further, preventing the model from learning patterns based on very few data points.
5. **Row Subsampling**: During training, only 80% of the training data was randomly selected for each boosting round, which helps prevent overfitting.
6. **Column Subsampling**: For each tree, 80% of the available features were randomly chosen, adding diversity and reducing overfitting.
7. **Minimum Loss Reduction**: A minimum improvement threshold of 0.1 was required to allow a split in the tree, reducing unnecessary complexity.
8. **Objective Function**: The model was optimized for binary classification, predicting the probability that a sample is defective.
9. **Evaluation Metric**: Model performance was evaluated using the Area Under the Curve, which is well-suited for imbalanced classification problems.
10. **Random Seed**: A fixed seed was set to ensure reproducibility of the results across runs.

5.2.2 Workflow

1. A command-line parser records file locations, fold settings, and output directories. The information is printed to the console and written to a log to enable traceability for each run.
2. The raw dataset is read into a DataFrame[37] and the columns are mapped to standard internal names for group, label, and signal channels.
3. If a cached feature file exists it is loaded; otherwise the raw sequences are passed to the feature routine described below, combined with numeric testing-machine parameters and DUT data as static features that are concatenated with sequence features, and written to cache for future runs.
4. The data is divided into training, validation, and test sets with StratifiedGroupKFold. Stratification is based on the defect label and the DUT 6 categorical static feature ensuring each fold keeps the same minority/majority ratio and a balanced distribution of DUT 6 categories.
5. The training portion is balanced with SMOTE-Tomek[38], adding minority class sequences while leaving validation and test sets untouched to prevent leakage.
6. XGBoost is trained with early stopping against the validation set. Hyper-parameters are tuned in a preliminary sweep and fixed to keep the search space stable across folds.
7. The fitted model is saved to a .pkl file[39] in a fold-specific directory so that later scripts can load it without retraining.

5.2.3 Feature Engineering

The feature extraction step converts each raw, variable-length univariate spatial sequence into a fixed-length feature vector suitable for XGBoost. All computations are carried out in a dedicated Python module built around the extract_statistical_features function. For every sequence we first apply a Savitzky–Golay[40] smoother to reduce measurement noise while preserving the overall shape. We then calculate ten types of sequence features that describe statistical, temporal, spectral, and geometric aspects of the signal. Scalar testing-machine parameter features and DUT data features are cleaned, imputed, and appended to the sequence features. The outcome is a fixed-width feature matrix that is cached on disk and reused during model training and evaluation.

The extracted features fall into nine broad categories:

1. Basic statistics summarise the amplitude distribution of each sequence, for example mean, variance, higher-order moments, percentiles, and inter-quartile range.
2. Time-domain variation captures short-term changes such as the number of zero crossings and the average absolute difference between consecutive samples.
3. Frequency-domain power spectrum measures include total spectral power, spectral entropy, dominant-frequency ratios, spectral centroid, roll-off, and related bandwidth statistics derived from the real FFT.
4. Complexity metrics quantify irregularity with approximate entropy, Katz fractal dimension, and Hjorth mobility and complexity.
5. Jerk descriptors describe the third finite difference of the signal by means, extremes, energy, and higher-order moments, together with a count of large jerk peaks.
6. Amplitude ratios look at percentiles and range, for example the ratio of the 25th to the 90th percentile and the full amplitude span.
7. Waveform shape factors characterise crest, impulse, shape, and clearance ratios based on peaks, root-mean-square, and absolute means.
8. Slope, asymmetry, and duration features describe how steeply the signal rises and falls between extrema, how symmetric each peak or valley is, and how long these segments last.
9. Cross-channel interactions, available when two channels are recorded, capture linear correlation, average phase difference, nonlinear coupling, and phase-difference entropy.

Together these descriptors encode amplitude, frequency content, complexity, geometric shape, and inter-channel relations of each unit in a single numeric vector. This representation works well with XGBoost and feeds directly into the resampling, training, and evaluation stages outlined above. Additionally, we implemented a custom feature extraction for peak and valley area features. This custom feature calculated the area between peaks and valleys in the univariate spatial sequences, which proved to be valuable for differentiating between normal and defect sequences. The peak and valley area function laid the foundation to some of the top performing sequence-based features according to SHAP evaluation. Our analysis showed that the standard deviation of valley peak areas strongly correlated with defect sequences, making this one of our most important features for classification

5.3 ConvTran Pipeline Implementation

The ConvTran codebase was forked from the publicly available repository by the original ConvTran paper. To make it suitable for our univariate spatial data, we adjusted the input embedding layer to accept single-channel sequences, added a mechanism for appending static feature tokens, and incorporated a focal-loss criterion to counter the class imbalance. Only the model checkpoint with the best validation loss was saved for each fold in the stratified k-fold cross validation. Checkpoints for alternative metrics such as minority class precision and minority F1-score were used but not included in the results as they output worse classification scores.

5.3.1 Static Feature Integration

In the main experiments, static features were included alongside the univariate spatial data. All available testing machine parameters and the, according to the company, most significant DUT features that describe the physical attributes of the heat exchangers were used. These features included both scalar and categorical variables. Categorical features were transformed using one-hot encoding[41] and were fit only on the training split, to prevent data leakage[42]. Static features were integrated into the model as a single token appended at the end of each sequence input. This allowed the transformer mechanism within ConvTran to attend to both the raw sequence and relevant static features[43].

5.3.2 Handling Class Imbalance with Focal loss

Focal loss replaces standard cross-entropy to address the large influence of the majority class by down-weighting easily classified examples. In each forward pass the predicted probability p_t of the true class is raised to a focusing power γ in the term $(1-p_t)^\gamma$, which amplifies the penalty on hard to classify, often minority-class, sequences. α balances the baseline importance of each class and is calculated automatically based on the class distribution.

5.3.3 Hyperparameter Optimization

The ConvTran model was tuned by hand, guided by transformer theory and empirical results rather than by an automated search. A focal loss focusing parameter of $\gamma = 3.0$ was selected because a stronger emphasis on hard-to-classify minority-class units helps counter the pronounced class imbalance in our data; preliminary runs with the standard value $\gamma = 2.0$ under-weighted minority errors and produced lower recall. We expanded the embedding size to $d_{\text{model}} = 64$ and set the feed-forward width to $d_{\text{ff}} = 256$, preserving the recommended $d_{\text{ff}}/d_{\text{model}} = 4$ ratio that supports stable gradient flow in transformer blocks[26]. Eight attention heads were retained so that each head processes a modest eight-dimensional subspace, which balances expressive power and memory cost on our variable-length sequences. Batch size and learning rate were left at the values proven stable in the original open-source ConvTran implementation[36], as tuning these parameters did not reveal consistent improvements. Training was capped at seventy epochs because validation loss barely improved after fifty epochs. 20 extra epochs gave us a margin if some occasional fold would converge more slowly. This configuration achieved the best minority class F1 score across our stratified group K-fold cross-validation runs and remained the top performer when we reran the procedure with different random seeds, which strengthens confidence that the chosen hyperparameters generalise beyond a single split.

5.4 Ensemble Model Development and Evaluation

To further improve the classification ability of our models, an ensemble approach was developed by combining the prediction probabilities of both ConvTran and XGBoost. The rationale was to leverage the wider probability distribution and high minority recall achieved by ConvTran, alongside the high-confidence, high-precision predictions produced by XGBoost. The observed distribution of predicted probabilities for ConvTran was broad, with many predictions in the intermediate range, whereas XGBoost tended to be very certain in its outputs, assigning probabilities either below 0.05 or above 0.95 for most samples. These statistical characteristics provided a basis for developing and testing a range of ensemble strategies.

5.4.1 Ensembling Strategies Explored

Several ensemble strategies were implemented to combine the predictions from ConvTran and XGBoost, each designed to take advantage of the models' distinct behaviors as reflected in their probability outputs.

5.4.2 Rule-Based Ensembles

A rule-based ensemble model was developed, defining logic for how to combine the two models' prediction probabilities by leveraging ConvTran's high-recall behavior when very confident and XGBoost's high precision otherwise:

1. If ConvTran probability > 0.9, predict the minority class since ConvTran's rare high-confidence outputs are almost always correct.
2. Else if XGBoost probability > 0.65 and ConvTran probability > 0.35, predict the minority class by combining XGBoost's precision with ConvTran's moderate support.
3. Else if XGBoost probability < 0.10 and ConvTran probability < 0.35, predict the majority class by reflecting both models' confidence in the negative outcome.
4. Else if XGBoost probability > 0.30 and ConvTran probability > 0.50, predict the minority class in cases where both models show some bias toward the minority class
5. Otherwise if the models are disagreement or both have low confidence, predict the majority class.

This hierarchy ensures that only when ConvTran is exceptionally confident does it unilaterally decide, while most predictions require mutual confidence—balancing ConvTran's recall strength against XGBoost's precision.

5.4.3 Weighted Average with Adaptive Thresholds

In this approach, a weighted average of the ConvTran and XGBoost probabilities was calculated, where the weights were selected as 0.6 for ConvTran and 0.4 for XGBoost. Rather than applying a fixed threshold to determine the final class, an adaptive threshold was used. Within uncertainty regions with probabilities between 0.45 and 0.7, the threshold was lowered to 0.45. Otherwise it stayed at the default 0.5.

5.4.4 Model-Switching Strategy

This ensemble approach selected the final prediction by relying on XGBoost over as long as XGBoost's confidence exceeded a threshold of 0.9 for minority class or fall below 0.1 for majority class. Otherwise, if ConvTran showed less confidence, ConvTran was allowed to make the prediction with a prediction probability threshold of 0.55.

5.4.5 Region-Based Ensemble

A more granular strategy was also implemented by dividing the joint probability space of ConvTran and XGBoost outputs into several regions. Each region had its own rule for determining the class and associated probability. High-agreement regions (both models confident in the same class) were handled directly, while disagreement or partial-confidence regions triggered tailored rules that prioritized either the more confident model or a default outcome. This method provided flexibility to design and test tailored ensemble logic for different joint probability scenarios.

6. Results

6.1 XGBoost Model Performance

The baseline XGBoost model with TSFresh[??] features demonstrated strong performance, achieving approximately 98% accuracy on normal class classification in test data, while reaching around 70% accuracy on the minority defect class.
We compared the baseline performance against several more advanced neural network architectures to determine if they could improve classification performance. Interestingly, despite their theoretical advantage for sequence modeling, the more complex neural architectures (CNNs, LSTMs, and transformers) did not outperform XGBoost with features from TSFresh based on F1-score[3] for the minority class for this dataset.
Performance analysis gave us some key insights:

1. XGBoost demonstrated superior classification of normal samples (98% accuracy) compared to all neural networks architectures that we tried.

2. While some ConvTran showed improvements in the classification of defect samples (∼90%) but a decrease in performance on classification of normal samples (∼93%), this apparent improvement in classification was misleading.

With our heavily imbalanced dataset (approximately 95% normal, 5% defect), standard accuracy metrics obscured the models actual performance.

6.1.1 Confidence-Based Classification Improvement

To further improve the XGBoost models F1-score on minority class, we tried a confidence based threshold strategy. This approach involves abstaining from classification when the models predicted probability falls within an uncertain range for example between 0.2 and 0.8, only making predictions when the model have high confidence.

Figure 2 illustrates how precision and recall for the minority class improve as the confidence threshold increases, alongside the corresponding coverage for both classes. Coverage represents the percentage of samples that receive a classification at each threshold level.

Several key observations can be made from this analysis:

1. Both precision and recall for the minority class show consistent improvement as the confidence threshold increases, with precision rising from approximately 0.70 at a threshold of 0.5 to over 0.90 at the highest thresholds.

2. The coverage for normal units remains high (above 0.95) until very high confidence thresholds, indicating that the model is consistently confident when classifying normal heat exchangers.

   The observed difference in coverage patterns between classes is likely attributable to the significant class imbalance in our dataset, comprising approximately 28,500 normal units (95%) and only 1,500 defect units (5%). This imbalance affects the model's ability to develop equally confident predictions for both classes, as it has substantially more examples of normal sequences to learn from during training.

3. However, the coverage for defect units decreases more rapidly as the threshold increases, indicating a tradeoff between classification quality and the proportion of defectt units that can be detected with high confidence.

4. At a confidence threshold of approximately 0.75, the model achieves a balanced performance with:

   - 0.78 precision for the minority class
   - 0.82 recall for the minority class

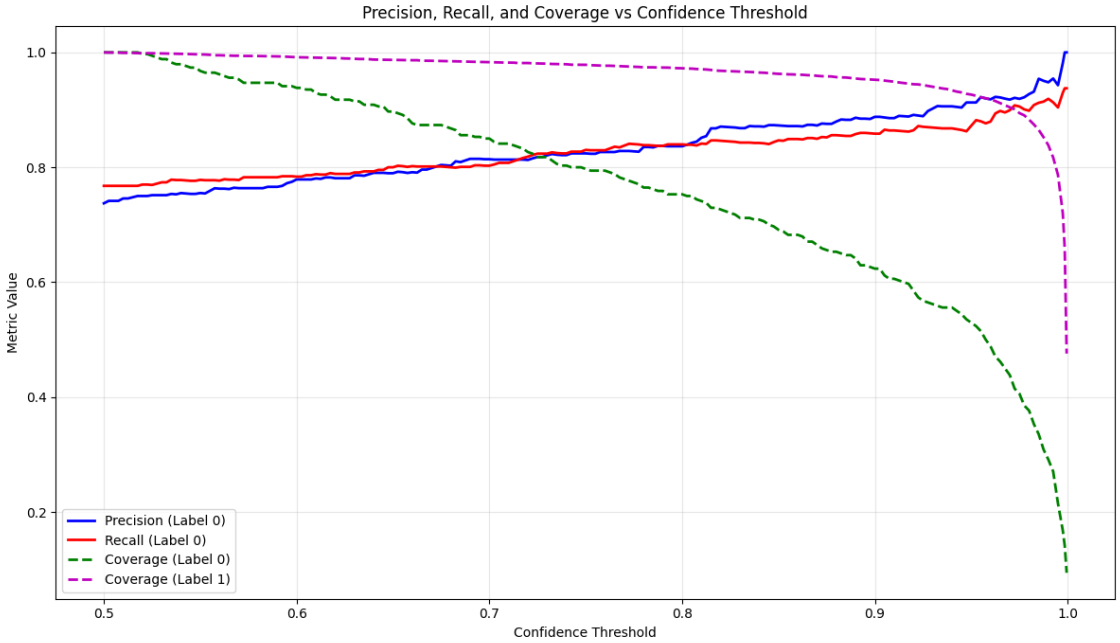

Figure 2: Precision, Recall, and Coverage vs Confidence Threshold. The graph shows how precision (blue) and recall (red) for the defect class increase as the confidence threshold rises, while coverage decreases for both defect (green) and normal (pink) classes.

- 0.80 coverage for minority class
- 0.97 coverage for majority class

This analysis demonstrates that implementing a confidence threshold allows for significant performance improvements without excluding an excessive number of units from classification. For industrial applications, this approach offers flexibility in balancing detection rates against false positives based on specific operational requirements. Furthermore, this confidence-based classification strategy could work well together with human supervision, where units with low-confidence predictions can be flagged for human expert review, while high-confidence predictions can proceed without intervention. Such a hybrid approach optimizes both automation efficiency and quality control in the manufacturing process.

6.1.2 5 fold cross validation

Table 1: Comparison of overall accuracy, minority class precision, recall, and F1 score for different model training strategies.

Model	overall accuracy	precision	recall	F1 score
Single split XGBoost	97,27%	73,03%	76,25%	74,61%
K fold cross validation XGBoost	97,16%	72,61%	73,79%	73,17%
Single split ConvTran Max Recall	92,17%	37,78%	90,01%	53,23%
Single split ConvTran	97.00%	72,24%	70,35%	71,28%
K fold cross validation ConvTran	96,78%	71,78%	65,37%	68,10%

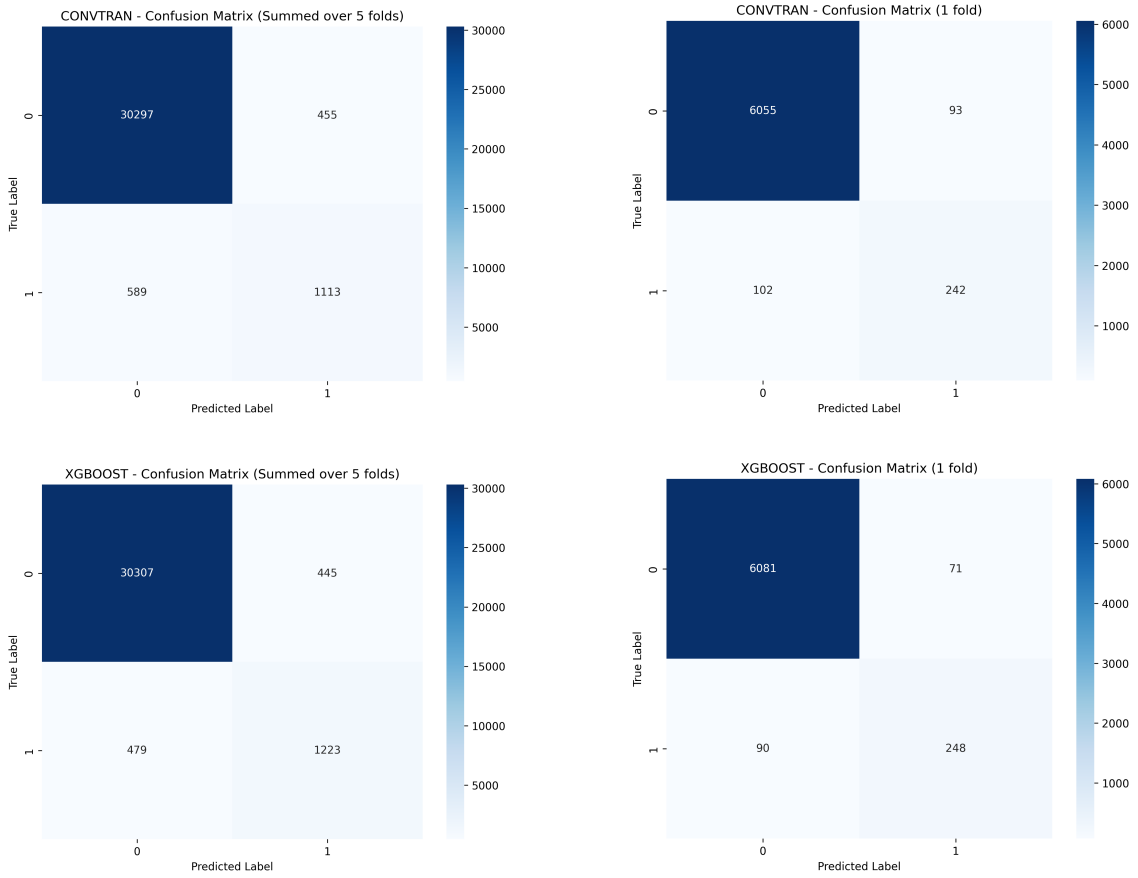

Figure 3: Confusion matrices for model evaluation. Top left: ConvTran (all 5-fold cross validation); top right: ConvTran (best single split); bottom left: XGBoost (all 5-fold cross validation); bottom right: XGBoost (best single split)

Figures 4–7 compare minority-class precision and recall for the three categorical values of DUT 6 (0, 10, 16).

1. **ConvTran**
   (a) Precision increases from 0.702 at value 0 to 0.851 at value 10, then falls to 0.613 at value 16 (Figure 4).
   (b) Recall follows the same ordering, moving from 0.659 at 0, to 0.720 at 10, and down to 0.583 at 16 (Figure 5).
2. **XGBoost**
   (a) Precision is equal to or higher than ConvTran in every category, registering 0.722, 0.829 and 0.613 for 0, 10 and 16 respectively (Figure 6).
   (b) Recall exceeds ConvTran at each value, coming in at 0.743, 0.762 and 0.638 (Figure 7).

Category 10 represents the easiest operating point for both models. XGBoost leads on both precision and recall, which limits false alarms while detecting more defective units. ConvTran displays wider confidence intervals, a sign of greater variability. The drop at value 16 highlights a setting that may benefit from closer process monitoring or targeted data augmentation.

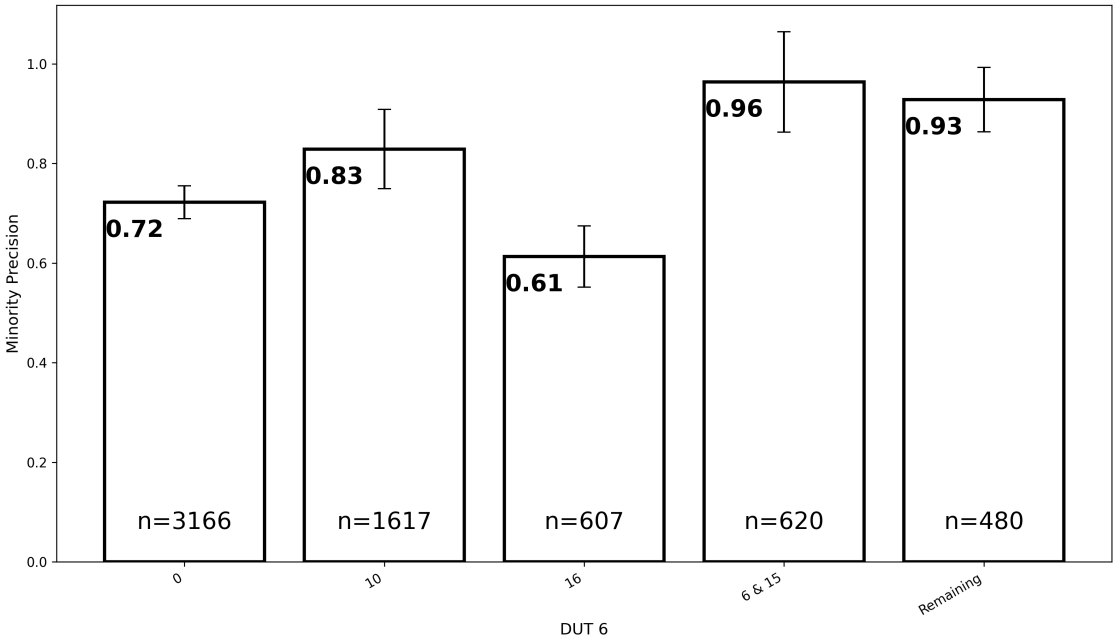

Figure 4: ConvTran's minority class precision on DUT 6

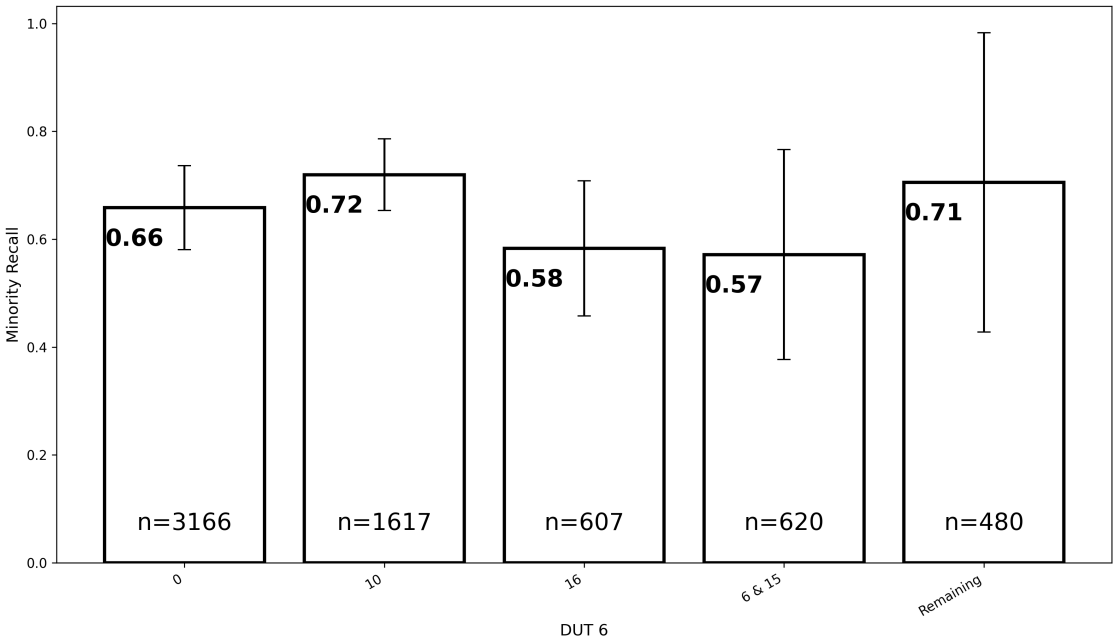

Figure 5: ConvTran's minority class recall on DUT 6

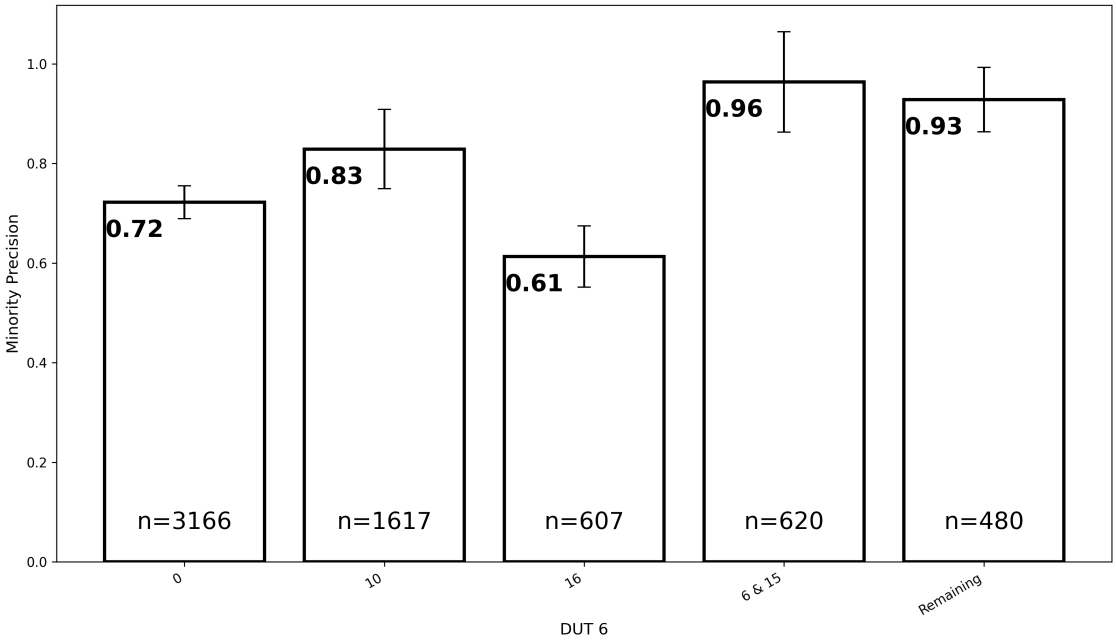

Figure 6: XGBoost's minority class precision on DUT 6

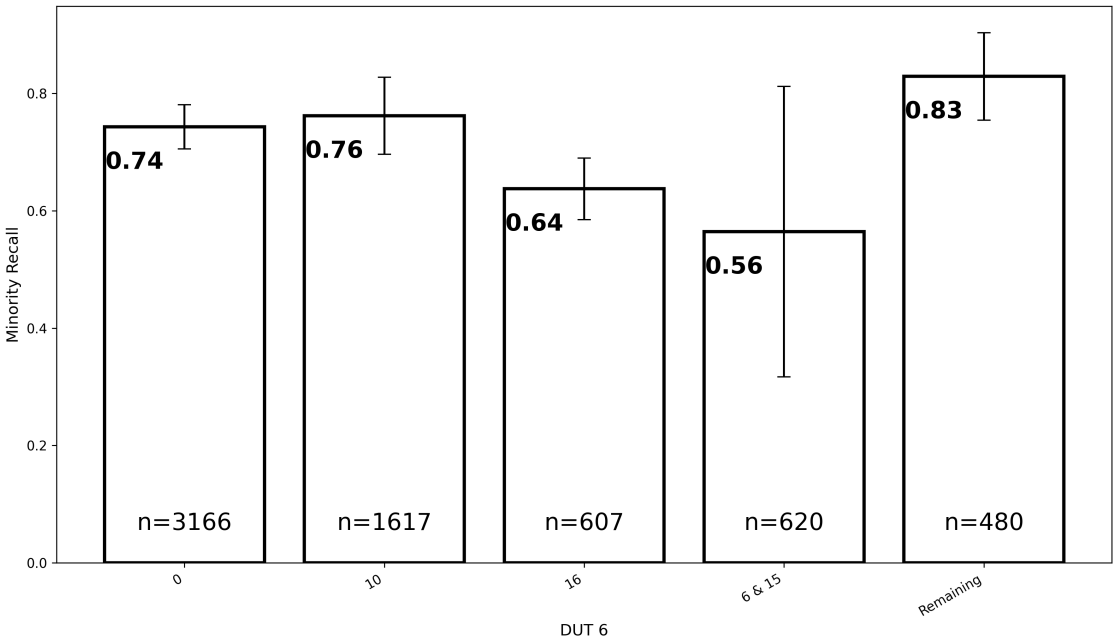

Figure 7: XGBboost's minority class recall on DUT 6

Figures 8–11 report precision and recall after grouping DUT 7 into four ranges.

1. **ConvTran**

   (a) Precision is strong in the Low bin (0.800) and highest in the High bin (0.853), dipping to 0.687 and 0.607 in the two mid-range bins (Figure 8).

   (b) Recall peaks at 0.768 in the Low bin, drops to 0.461 in the Upper-mid bin, and recovers to 0.562 in the High bin (Figure 9).

2. **XGBoost**

   (a) Precision records 0.817 in Low, 0.728 in Lower-mid, 0.572 in Upper-mid and 0.781 in High (Figure 10).

   (b) Recall shows 0.842, 0.722, 0.523 and 0.712 across the same bins (Figure 11).

Size has a big impact on performance . Both models classify the smallest units well, with precision above 0.80 and XGBoost recall above 0.84. The 46–58 range is the most difficult; recall falls below 0.55 for both models. For the largest units, ConvTran attains its best precision while XGBoost maintains the higher recall. The complementary error patterns suggest that an ensemble or adaptive thresholding could merge these strengths. The two mid-range bins should receive priority for extra data or process checks because they are the hardest to classify.

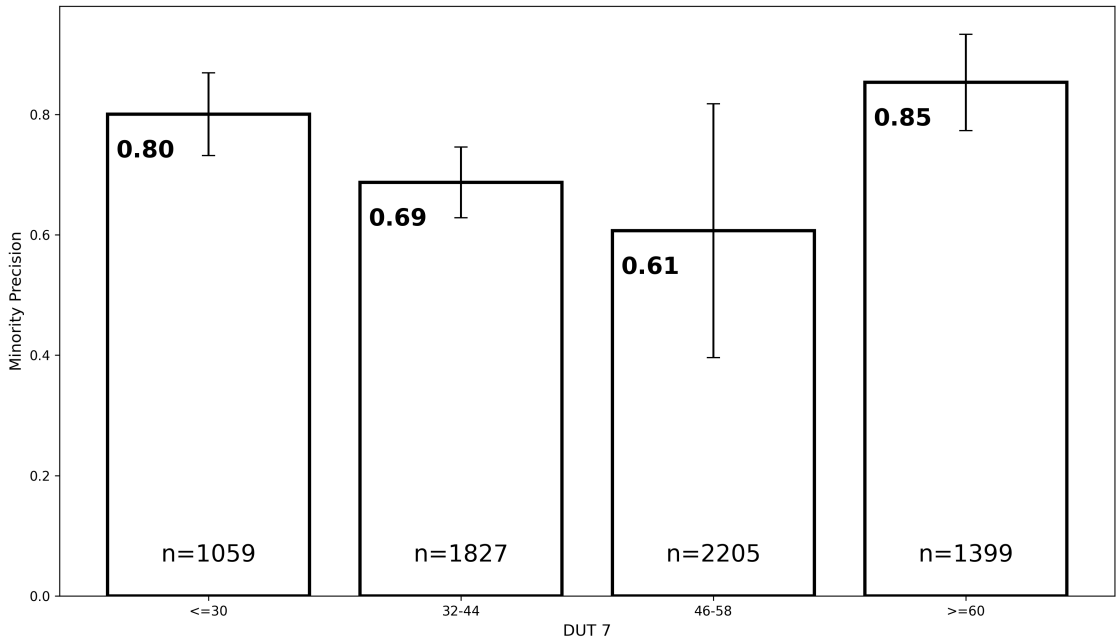

Figure 8: ConvTran's minority class precision on DUT 7

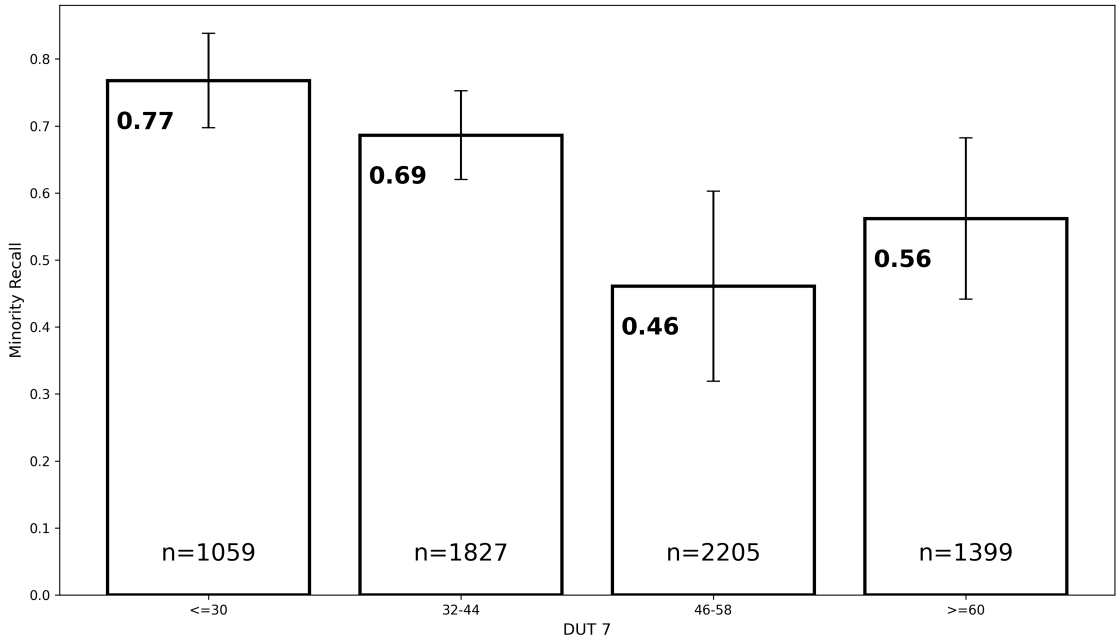

Figure 9: ConvTran's minority class recall on DUT 7

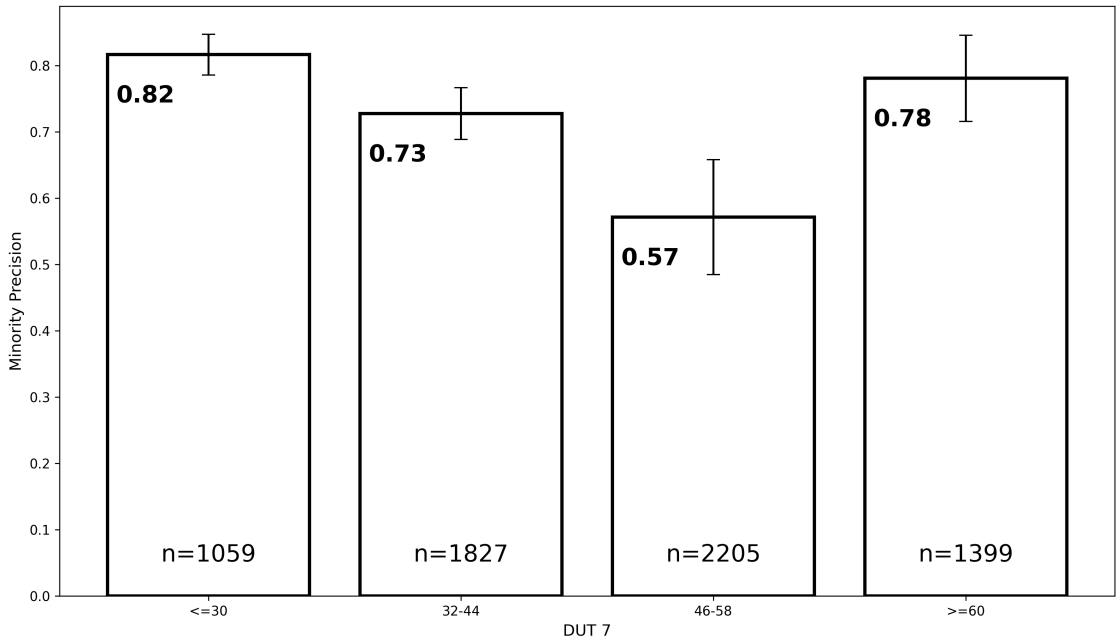

Figure 10: XGBboost's minority class precision on DUT 7

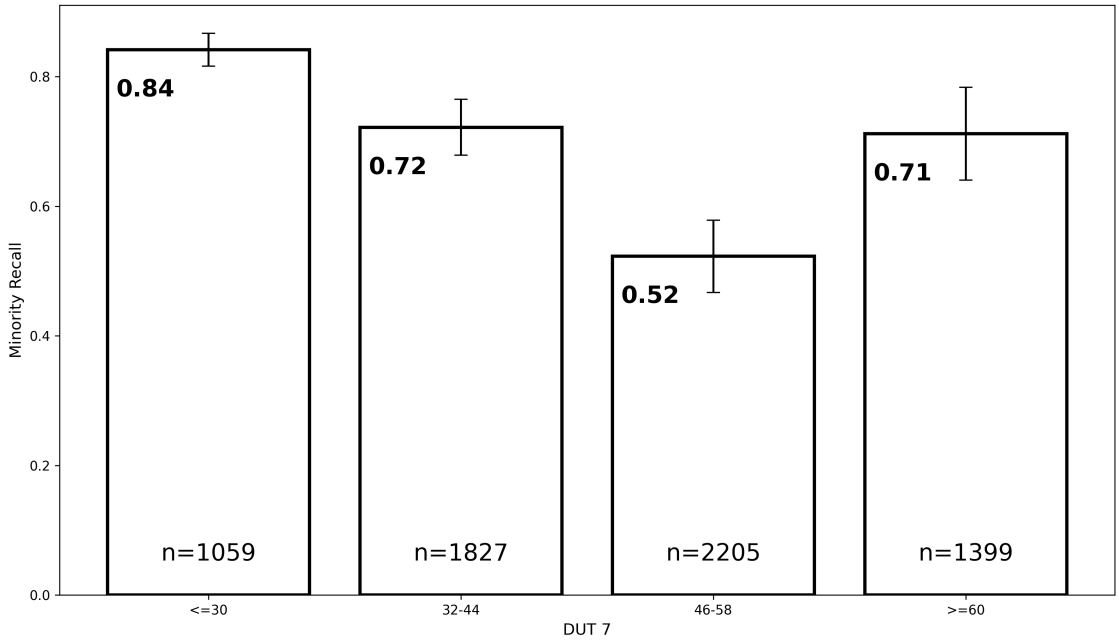

Figure 11: XGBboost's minority class recall on DUT 7

6.2 Ensemble Model Performance

The best fold of both models were extracted of it's prediction probability and combined with 4 different ensemble models. Figures **??** and shows the high confidence of our XGBoost model and the more evenly distributed confidence of our ConvTran model that was combined to multiple high performing ensemble models.

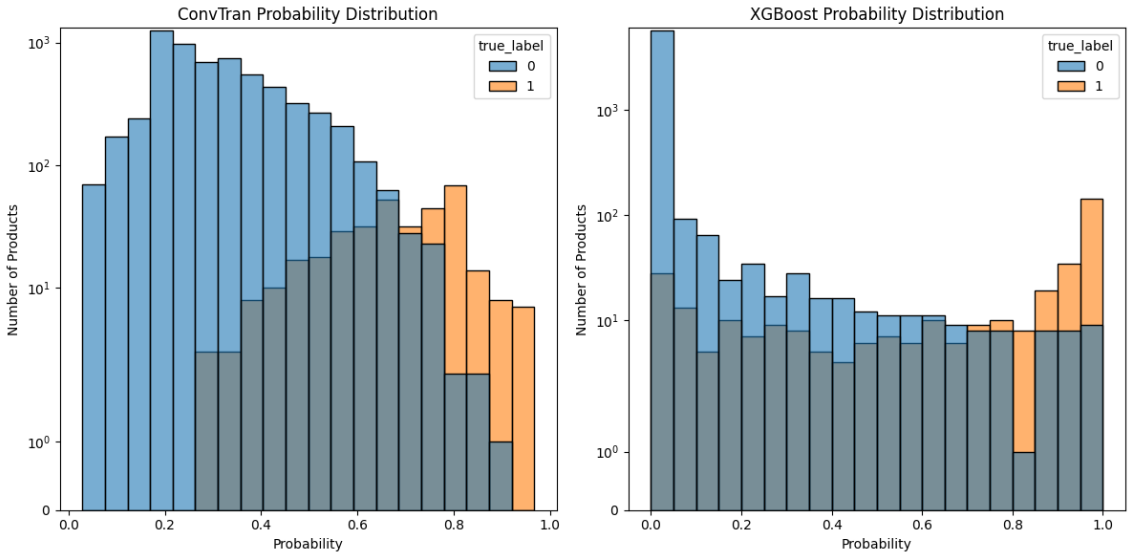

Figure 12: Log distribution of prediction probability

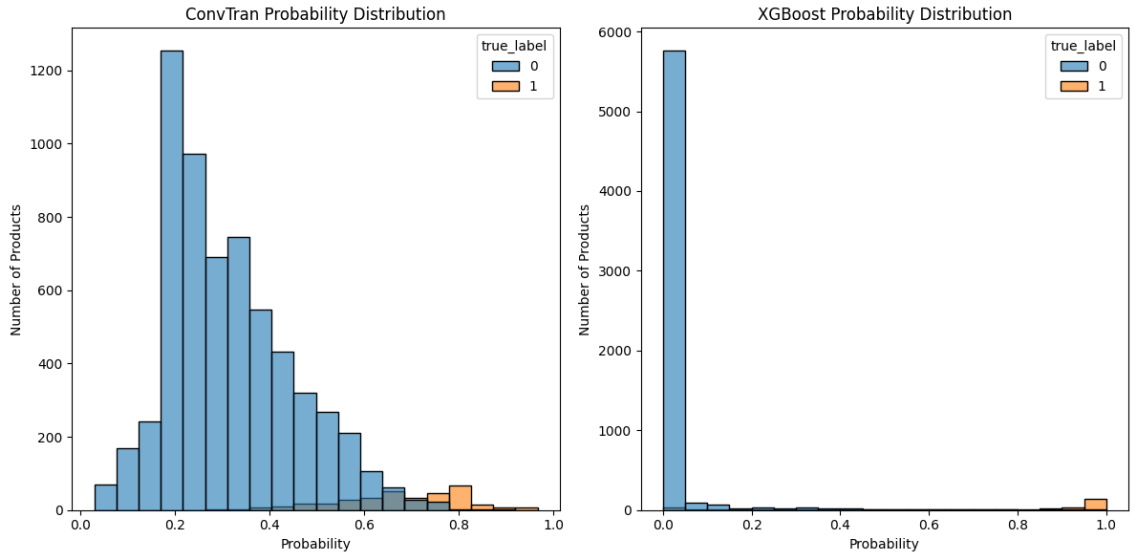

Figure 13: Absolut distribution of prediction probability

Table 2 shows the mean test accuracy, minority class precision, minority class recall and F1 score for the two base learners and the four ensemble variants.

Table 2: Performance of the ensemble models. Precision, recall and F1 score is evaluated based on the minority class.

Model	Accuracy	precision	recall	F1
Rule-Based	0.959	0.569	0.928	0.705
Weighted adaptive	0.984	0.819	0.902	0.858
Model-switching	0.986	0.889	0.835	0.861
Region-based	0.979	0.788	0.827	0.807

6.2.1 Rule-based ensemble

This variant delivers the strongest minority recall at 0.928 by giving precedence to confident ConvTran signals and lowering the decision threshold in disagreement zones. The gain in recall is offset by a fall in precision to 0.569 and a moderate reduction in accuracy, outcomes that translate to more false alarms but fewer missed defects. Such behaviour is suitable when the factory wishes to minimise the risk of shipping a faulty unit and can accept extra inspections.

6.2.2 Weighted adaptive ensemble

The weighted average with adaptive thresholds achieves a balanced profile. Accuracy rises to 0.984 and minority precision to 0.819 while recall remains high at 0.902, producing an F1 score of 0.858. If the weighted average is in an uncertain region, the lowered threshold keeps the recall high.

6.2.3 Model-switching ensemble

Switching between the base learners according to their confidence yields the highest accuracy (0.986) and minority precision (0.889). Minority recall settles at 0.835, lower than the rule-based or weighted approaches. This approach is ideal for production lines with limited inspection capacity where minimizing false alerts is critical.

6.2.4 Region-based ensemble

Partitioning the probability space into fine-grained regions provides a middle ground. Accuracy reaches 0.979 with minority precision of 0.788 and recall of 0.827, giving an F1 score of 0.807. The

method preserves most of the sensitivity gain over XGBoost while containing the false-positive count better than the rule-based alternative.

Combining prediction probability from two complementary models improves every metric compared with the single learners. The rule-based ensemble maximizes minority recall, the model-switching strategy secures the strictest minority precision and highest accuracy, and the weighted adaptive ensemble offers the most even balance across metrics. The region-based variant performed the worst but still falls between these extremes. When avoiding missed defects is the most important, the rule-based approach is preferable. When limiting false alarms takes priority, model-switching is the safer choice.

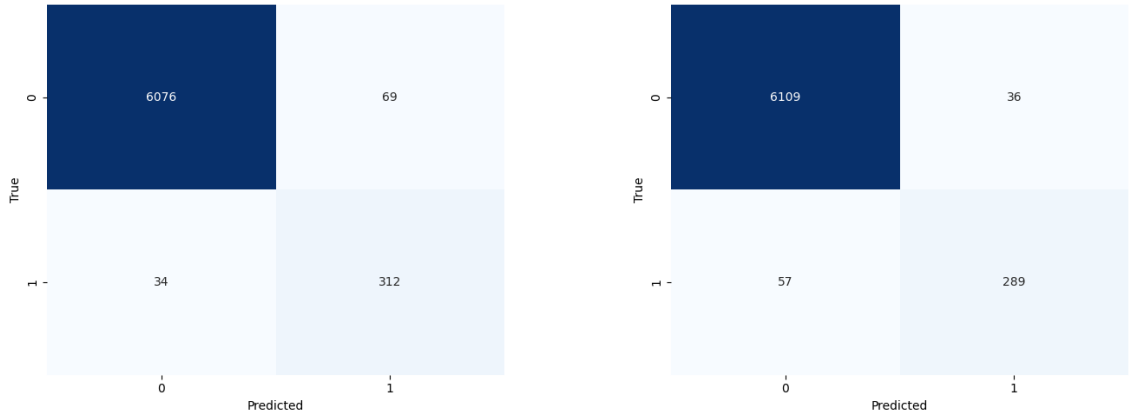

Figure 14: Confusion matrices for the weighted adaptive ensemble (left) and the model-switching ensemble (right).

7. Discussion

This study aimed to develop a supervised machine learning model that could differentiate between normal and defect heat exchangers based on distance series data. Our results demonstrate that XGBoost outperformed our deep learning method ConvTran on both minority precision and overall minority f1 score. ConvTran achieved a higher minority recall, and in combination with the fact that the two models had very few common false negatives and false positives, we could use both models' prediction probabilities for an ensemble model. In this section we will interpret our findings, discuss their implications, and examine how they could relate to our research questions.

7.1 RQ1: Interpretation of single Model Performance

7.1.1 XGBoost vs. Neural Network Approaches

One of the most notable findings was that our XGBoost model both with simple to use TSFresh features and more custom features outperformed more complex neural network architectures in terms of F1-score on the minority class, as evaluated through 5 fold cross-validation. While ConvTran achieved higher recall for the minority class (90% vs 70%), this came at the cost of more false positives, as seen in this figure. This trade-off significantly impacts the practical usage of these models in manufacturing environments.

These results addresses our first research question about comparative model performance, though with some important considerations regarding reliability. Our cross-validation approach provides some confidence in the results, but there might be some factors limiting the robustness of our findings. One example would be our preprocessing pipeline, while designed to avoid data leakage, it may still contain biases due to our limited understanding of the underlying manufacturing process. The superior performance of XGBoost aligns with recent findings by Farahani et al. [2] and the benchmarking study by Florek and Zagdanski [3], which highlighted the strong performance of gradient boosting algorithms for classification tasks on structured tabular data. Our results suggest that for this data, the feature engineering with XGBoost captures the relevant patterns more effectively than the neural architectures.

7.1.2 Performance Across Different Data Categories

The significant class imbalance in our dataset presented a major challenge for model development. Our analysis of different approaches to handle this imbalance revealed several insights:

1. The use of focal loss in ConvTran training proved effective for maintaining high recall for the minority class, aligning with findings by Lin et al. [12] on the benefits of this approach for imbalanced classification problems.

2. The resampling techniques (undersampling and SMOTE [44]) applied to the XGBoost model helped minimize the bias toward the majority class, though each introduced its own possible trade-offs in terms of information loss vs potential overfitting.

3. The confidence-based classification approach demonstrated a particularly promising way to balance precision and recall for practical applications, allowing for flexible threshold adjustment based on specific operational requirements.

This confidence-based approach deserves particular attention as it represents a practical solution for production environments. By adjusting the confidence thresholds according to specific operational requirements, manufacturers can effectively balance the costs of false positives (normal units classified as defective) and false negatives (defective units classified as normal). This flexibility is crucial in industrial settings where the relative costs of these error types vary significantly based on product specifications, customer requirements, and production constraints. Our research shows that increasing the confidence threshold consistently improves both precision and recall on both the majority and minority class, allowing manufacturers to make a deliberate trade-off between classification quality and coverage. This flexibility enables quality control systems to adjust according to specific priorities like more coverage or confidence in the models predictions, making

it especially valuable for industrial applications where the cost of different error types may vary across products and customer requirements.

7.2 RQ2: Interpretation of ensemble Model Performance

The second research question asked whether an ensemble that blends the probability outputs of ConvTran and XGBoost could improve the detection of defective units in our univariate spatial data. The results in Table 2 verify that it can. Among the variants, the weighted adaptive average reached an accuracy of 0.984 with minority-class precision 0.819, recall 0.902, and F_1 0.858, while the model-switching rules achieved an accuracy of 0.986 with precision 0.889, recall 0.835, and F_1 0.861. Both scores are well above those of the individual learners, whose F_1 values were 0.68 for ConvTran and 0.73 for XGBoost, which answers RQ2. The improvement stems from the fact that the two base models make different mistakes: on the 6491 unit test set they shared only 14 false negatives and 11 false positives. When the ensemble merges the calibrated probabilities it recovers many of the defects that either model would miss alone, raising minority performance without a large increase in false alarms.

Between the two leading ensembles, the weighted adaptive average favours recall. It keeps the share of missed defects low and requires only two tuning parameters, which makes it a safe default for most production lines. The model-switching rules favour precision. They reduce unnecessary re-inspections and suit factories where inspection capacity is tight or false alarms are costly, though their two confidence cut-offs will need retuning if the testing-machine parameters shift. A sensible deployment plan is therefore to start with the weighted adaptive scheme and move to the switching logic if the line becomes overloaded by extra reviews. Several limitations must be noted. The experiments used a single temporal snapshot; wider cross-validation is needed for stronger statistical evidence. Thresholds were chosen by manual inspection, so an automated search may discover better trade-offs. Running two full pipelines and a fusion layer adds latency that must be profiled on the edge hardware attached to the testing machine. Our findings echo earlier work by Mienye and Sun, who show that combining heterogeneous learners can raise robustness on imbalanced sensor data. This study extends that insight to variable-length sequences and confirms it on a real manufacturing dataset. Because class imbalance with roughly five percent defects is common in weld-seam ultrasound, pipeline pressure tests, and similar inspection tasks, the approach should generalise. Only the confidence thresholds and, if necessary, the weight parameter need recalibration when new data arrive. In summary, the ensemble that merges ConvTran and XGBoost probabilities answers RQ2 in the affirmative: it improves minority-class F_1 over either base learner on its own. The weighted adaptive average offers the best starting point, while the switching rules provide a precision-oriented alternative when inspection resources are scarce.

7.3 Practical Implications for Industrial Quality Control

From an industrial perspective, our findings have several significant implications for heat exchanger quality control at the manufacturer:

1. The developed models demonstrate that automated defect detection using machine learning is feasible and effective, potentially reducing the need for manual inspection and improving quality consistency.

2. The confidence-based classification approach offers a practical framework for integrating these models into the existing quality control process, where high-confidence predictions can proceed automatically while uncertain cases can be flagged for expert review.

3. The performance variations across different heat exchanger configurations (as indicated by the DUT features) suggest that specialized models for specific product lines or configurations might further improve detection accuracy.

4. The high performance of feature-based approaches indicates that explicit feature engineering remains valuable in industrial settings, even as deep learning advances.

7.4 Limitations and Future Work

Despite the promising results, this study has several limitations that should be acknowledged:

1. **Changes over time**: The current study does not address potential changes in defect patterns over time due to manufacturing process changes. Future work should explore how these models perform over extended periods and whether they require periodic retraining.
2. **Explainability challenges**: While XGBoost offers some level of interpretability through feature importance, the ConvTran model and ensemble approaches are less transparent. For industrial applications, developing more explainable models that can pinpoint specific issues would be valuable.
3. **Computational requirements**: The feature extraction process using custom features generates approximately 150 features per sequence, which may become computationally expensive for real-time applications in low performance edge devices. Future work could focus on further feature selection to identify the most discriminative subset of features.

Several avenues for future research emerge from these limitations:

1. Exploration of multimodal approaches that incorporate additional sensor data or product specifications beyond the current feature set.
2. Investigation of model explainability techniques specifically tailored for classification in low dimensional spatial data in manufacturing contexts.
3. Extension of the confidence-based classification approach to incorporate economic cost models that explicitly balance the costs of false positives against false negatives.

8. Conclusions

This study successfully developed and evaluated supervised machine learning models for defect detection in heat exchangers using univariate spatial data from the manufacturing process. Our research addressed two primary research questions regarding comparative model performance and the effectiveness of ensemble methods for improving classification accuracy in industrial control applications.

8.1 Key Findings

8.1.1 Research Question 1: Comparative Model Performance

Our analysis revealed that XGBoost meaningfully outperformed the ConvTran neural network architecture in terms of minority class F1-score (0.73 vs 0.68) and precision, while ConvTran achieved superior recall (90% vs 70%). This finding suggests that architectural complexity doesn't guarantee better performance, particularly with limited training data. With approximately 30 000 samples where only 1 500 are defects, the superior performance of XGBoost likely reflects the efficiency of gradient boosting methods when training with limited data, whereas neural networks typically require larger datasets generalize well to unseen data.

It is reasonable to expect that over time as more data is collected, the ConvTran architecture might eventually surpass the feature-engineered XGBoost models. The good performance of XGBoost aligns with recent findings by Farahani et al and the benchmarking study by Florek and Zagdanski, demonstrating that feature engineering combined with gradient boosting remains highly competitive for structured tabular data even when we derive it from univariate spatial data.

The effectiveness of our stratified cross-validation approach provided a good performance estimate despite the significant class imbalance present in our dataset. The confidence-based classification approach demonstrated particular promise for industrial applications, allowing flexible threshold adjustment based on operational requirements while maintaining high performance metrics.

8.1.2 Research Question 2: Ensemble method Effectiveness

The ensemble approaches answer our seconds research question. Both the weighted adaptive ensemble (F1: 0.858) and model-switching ensemble (F1: 0.861) outperformed individual models, with an improvement of approximately 18% over the best single model. The ensemble success stemmed from the complementary nature of the based models predictions, with only 14 shared false negatives and 11 shared false positives across 6491 test samples.

This low error overlap between ConvTran and XGBoost demonstrates that the models make different types of mistakes on data that is hard to classify, making them suitable for ensemble combination. The weighted adaptive ensemble offers the best balance for general use, while the model-switching approach provides superior precision.

8.2 Industrial Impact and Practical Significance

The developed models demonstrate clear practical value for heat exchanger quality control:

1. **Automated Detection Feasibility**: Our results demonstrate that machine learning-based defect detection is feasible and effective for this industrial application. However, even with high- performance models achieving F1-scores over 0.9 through confidence-based thresholding, fully automated detection will not be possible as long as the model isn't confident for all units. Nevertheless, these models could provide significant value as intelligent screening tools that can prioritize units for manual inspection, automatically approve high-confidence normal units, and assist human inspectors by flagging potential defects, possibly improving both quality control efficiency and consistency.

2. **Flexible Deployment Options**: The confidence-based classification approach provides a practical framework for integrating into the current quality control processes, allowing for high-confidence predictions that can help manual inspectors make more confident decisions.

8.3 Study Limitations and Future Directions

1. **Stability**: The current study only used the data up to this point in time. Long-term deployment would require validation of model performance stability over time and potential retraining.

2. **Generalizability**: While our approach shows promise for this data, broader validation with future data would strengthen or disprove generalizability claims.

8.4 Final Conclusions

This study succesfully demonstrates that supervised machine learning can provide effective defect detection for heat exchanger manufacturing quality control. The combination of feature-engineered XGBoost models with the ConvTran architecture in ensemble configurations offers a solution that balances performance, interpretability, and practical deployment considerations.

The research contributes to the growing body of evidence that hybrid approaches combining traditional machine learning with modern deep learning techniques can outperform either approach alone, particularly in industrial applications with limited data and strict performance requirements. Our confidence-based classification framework provides a valuable template for integrating such systems into production environments while maintaining human oversight capabilities.

References

[1] V. Chandola, A. Banerjee and V. Kumar, 'Anomaly detection: A survey,' *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[2] M. A. Farahani, M. McCormick, R. Harik and T. Wuest, 'Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms,' *Robotics and Computer-Integrated Manufacturing*, vol. 91, p. 102 839, 2025.

[3] P. Florek and A. Zagdanski, 'Benchmarking state-of-the-art gradient boosting algorithms for classification. arxiv 2023,' *arXiv preprint arXiv:2305.17094*, 2023.

[4] H. He and E. A. Garcia, 'Learning from imbalanced data,' *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[5] M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks,' *Neural networks*, vol. 106, pp. 249–259, 2018.

[6] T. Saito and M. Rehmsmeier, 'The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,' *PloS one*, vol. 10, no. 3, e0118432, 2015.

[7] A. de Giorgio, G. Cola and L. Wang, 'Systematic review of class imbalance problems in manufacturing,' *Journal of Manufacturing Systems*, vol. 71, pp. 620–644, 2023.

[8] S. M. I. Walker, *F-score: What are accuracy, precision, recall, and f1 score?* 2023. [Online]. Available: https://klu.ai/glossary/accuracy-precision-recall-f1.

[9] H. White, *Computer vision basics: Confidence accuracy*, 2025. [Online]. Available: https://www.leverege.com/blogpost/computer-vision-basics-how-confidence-accuracy-and-thresholds-impact-performance.

[10] J. Brownlee, *Random oversampling and undersampling for imbalanced classification*, 2021. [Online]. Available: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/.

[11] G. E. Batista, R. C. Prati and M. C. Monard, 'A study of the behavior of several methods for balancing machine learning training data,' *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, 'Focal loss for dense object detection,' in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[13] A. Nawaz, Z. Huang, S. Wang, A. Akbar, H. AlSalman and A. Gumaei, 'Gps trajectory completion using end-to-end bidirectional convolutional recurrent encoder-decoder architecture with attention mechanism,' *Sensors*, vol. 20, no. 18, p. 5143, 2020.

[14] sktime developers, *TSInterpolator: Univariate spatial data resampler based on* `scipy.interpolate.interp1d`, https://www.sktime.net/en/v0.37.0/api_reference/auto_generated/sktime.transformations.panel.interpolate.TSInterpolator.html, accessed 24 May 2025, 2025.

[15] M. Christ, N. Braun, J. Neuffer and A. W. Kempa-Liehr, 'Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package),' *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[16] X. Zhang *et al.*, 'A survey on data preparation and feature engineering in machine learning,' *Research Paper*, 2020. [Online]. Available: https://www.academia.edu/42907535/A_Survey_on_Data_Preparation_and_Feature_Engineering_in_Machine_Learning.

[17] 'Deep learning for time series forecasting: A survey,' *International Journal of Machine Learning and Cybernetics*, 2025. DOI: 10.1007/s13042-025-02560-w.

[18] 'Opening the black box: Interpretable machine learning for predictor finding of metabolic syndrome,' *BMC Endocrine Disorders*, 2022. DOI: 10.1186/s12902-022-01121-4.

[19] J. H. Friedman, 'Greedy function approximation: A gradient boosting machine,' *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. DOI: 10.1214/aos/1013203451. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full.

[20] T. Chen and C. Guestrin, 'XGBoost: A scalable tree boosting system,' in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: ACM, 2016, pp. 785–794, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. [Online]. Available: https://doi.org/10.1145/2939672.2939785.

[21] R. Tibshirani, 'Regression shrinkage and selection via the lasso,' *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. DOI: 10.1111/j.2517-6161.1996.tb02080.x. [Online]. Available: https://academic.oup.com/jrsssb/article/58/1/267/7027929.

[22] A. E. Hoerl and R. W. Kennard, 'Ridge regression: Biased estimation for nonorthogonal problems,' *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970. DOI: 10.1080/00401706.1970.10488634. [Online]. Available: https://www.jstor.org/stable/1267351.

[23] R. Kohavi *et al.*, 'A study of cross-validation and bootstrap for accuracy estimation and model selection,' in *Ijcai*, Montreal, Canada, vol. 14, 1995, pp. 1137–1145.

[24] I. D. Mienye and Y. Sun, 'A survey of ensemble learning: Concepts, algorithms, applications, and prospects,' *Ieee Access*, vol. 10, pp. 99 129–99 149, 2022.

[25] R. Kundu, *The essential guide to ensemble learning*, https://www.v7labs.com/blog/ensemble-learning-guide, Accessed: 27 May 2025, Jan. 2024.

[26] A. Vaswani, N. Shazeer, N. Parmar *et al.*, 'Attention is all you need,' *Advances in neural information processing systems*, vol. 30, 2017.

[27] A. Dosovitskiy, L. Beyer, A. Kolesnikov *et al.*, 'An image is worth 16x16 words: Transformers for image recognition at scale,' *arXiv preprint arXiv:2010.11929*, 2020.

[28] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty and C. Eickhoff, 'A transformer-based framework for multivariate time series representation learning,' in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 2114–2124.

[29] N. M. Foumani, C. W. Tan, G. I. Webb and M. Salehi, 'Improving position encoding of transformers for multivariate time series classification,' *Data mining and knowledge discovery*, vol. 38, no. 1, pp. 22–48, 2024.

[30] L. Wachowiak, P. Tisnikar, A. Coles, G. Canal and O. Celiktutan, 'A time series classification pipeline for detecting interaction ruptures in hri based on user reactions,' in *Proceedings of the 26th International Conference on Multimodal Interaction*, 2024, pp. 657–665.

[31] Z. Deng, Y. Zhang, L. Zhang and J. Cong, 'A transformer and random forest hybrid model for the prediction of non-metallic inclusions in continuous casting slabs,' *Integrating Materials and Manufacturing Innovation*, vol. 12, no. 4, pp. 466–480, 2023.

[32] A. Haque and H. Soliman, 'A transformer-based autoencoder with isolation forest and xgboost for malfunction and intrusion detection in wireless sensor networks for forest fire prediction,' *Future Internet*, vol. 17, no. 4, p. 164, 2025.

[33] E. Sylligardos, P. Boniol, J. Paparrizos, P. Trahanias and T. Palpanas, 'Choose wisely: An extensive evaluation of model selection for anomaly detection in time series,' *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 3418–3432, 2023.

[34] N. M. Foumani and S. Shumaly, *ConvTran: Improving position encoding of transformers for multivariate time series classification*, https://github.com/Navidfoumani/ConvTran, GitHub repository, accessed 27 May 2025, 2023.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, *StratifiedGroupKFold — scikit-learn api reference*, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedGroupKFold.html, Scikit-learn, accessed 27 May 2025, 2024.

[36] N. M. Foumani, *Convtran: A pytorch implementation of improving position encoding of transformers for multivariate time series classification*, https://github.com/Navidfoumani/ConvTran, Accessed: 2025-05-15, 2023.

[37] The pandas development team, *pandas.DataFrame — pandas documentation*, https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html, Accessed: 27 May 2025, 2024.

[38] G. Lemaıtre, F. Nogueira and C. K. Aridas, *SMOTETomek— imbalanced-learn api reference*, https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTETomek.html, Accessed 27 May 2025, 2024.

[39] P. S. Foundation, *pickle— python object serialization*, https://docs.python.org/3/library/pickle.html, Accessed: 27 May 2025, 2024.

[40] SciPy Community, *scipy.signal.savgol_filter — scipy documentation*, https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html, Accessed 27 May 2025, 2024.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, *OneHotEncoder — scikit-learn api reference*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html, Accessed 27 May 2025, 2024.

[42] A. Apicella, F. Isgrò and R. Prevete, 'Don't push the button! exploring data leakage risks in machine learning and transfer learning,' *arXiv preprint arXiv:2401.13796*, 2024.

[43] Y. Gorishniy, I. Rubachev, V. Khrulkov and A. Babenko, 'Revisiting deep learning models for tabular data,' *Advances in neural information processing systems*, vol. 34, pp. 18 932–18 943, 2021.

[44] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, 'Smote: Synthetic minority oversampling technique,' *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.