ROCKET-LRP: Explainable Time Series Classification with Application to Anomaly Prediction in Manufacturing

Zhijian Ling
zhijian.ling@mail.utoronto.ca
University of Toronto
Toronto, Ontario, Canada

Takuya Aoyama
takuya.aoyama@konicaminolta.com
Konica Minolta
Mississauga, Ontario, Canada

Keijiro Yano
keijiro.yano@konicaminolta.com
Konica Minolta
Mississauga, Ontario, Canada

Eldan Cohen
eldan.cohen@utoronto.ca
University of Toronto
Toronto, Ontario, Canada

Abstract

Time Series Classification is a popular approach in machine learning with many applications. The Random Convolutional Kernel Transform (ROCKET) model has achieved state-of-the-art performance in various time-series classification tasks due to its ability to capture complex patterns and temporal relationships. However, its reliance on random convolutions hinders the explainability of the model, as the relationships between the transformed features and the original input data become obscured. To address these challenges, we propose a novel approach for computing explanations in ROCKET-based time-series classification models that integrates Layer-wise Relevance Propagation with either model-agnostic post-hoc or model-intrinsic local explanation techniques. We implement our approach for two widely used classification models and three local explanation techniques. We validate our approach on two simulated datasets, demonstrating its faithfulness and effectiveness. Additionally, we present an application of our approach to anomaly prediction in real-world manufacturing data and show that it provides superior local explanations compared to popular explanation techniques such as SHAP and LIME.[1]

CCS Concepts

• **Computing methodologies → Machine learning**; **Learning paradigms**; **Supervised learning**; *Supervised learning by classification*;

Keywords

Post-hoc explanation, Time-series classification, Sensor data, Manufacturing

1 Introduction

Machine learning has become a powerful tool across domains, including finance [27], healthcare [10], supply chain management [23], and process mining [28]. In manufacturing, it plays a crucial role in process optimization [8], quality control [17], and anomaly prediction [32]. By building complex relationships with data, machine learning techniques allow manufacturers to enhance product quality while reducing reliance on costly and time-consuming testing [33].

Manufacturing processes are inherently time-dependent, with products moving through a pipeline where characteristics are recorded at each step. These steps are highly interdependent, and deviations in one stage can propagate through the pipeline, affecting the final product [18]. For example, the number of scratches or foreign matter in a product is closely tied to the conditions on the manufacturing line. Capturing these temporal dependencies is critical for understanding and improving manufacturing processes.

Explainability in machine learning is essential for ensuring trust and transparency, and it is now becoming ubiquitous across various domains [48]. In manufacturing, explainability is particularly crucial, allowing factory managers to understand the reasoning behind a model's predictions [1]. Explainable Time Series Classification models are vital for detecting anomalies and identifying defects, where actionable insights can help prevent costly errors and support informed decision-making [46].

Convolutional models effectively capture temporal patterns in time-series data, making them valuable for time-series applications, including manufacturing [47]. ROCKET, which combines features extracted from randomly initialized convolutional kernels with a conventional predictive model, has achieved state-of-the-art performance in various time-series tasks [42]. However, despite its strong empirical success, ROCKET's reliance on random convolutions poses challenges to its explainability.

In this paper, we propose a novel approach to explain the predictions of ROCKET-based time-series classification models by combining Layer-wise Relevance Propagation (LRP) [6] with local explanation techniques. Our work aims to bridge the gap between the accuracy of ROCKET-based models in TSC tasks and the need for model explainability. We make the following contributions:

- We introduce ROCKET-LRP, a novel approach for integrating LRP with post-hoc or intrinsic local explanations to provide explanations for the predictions of ROCKET-based TSC models.
- We implement our approach for two widely used classification models, XGBoost and Logistic Regression (LR), and three local explanation techniques (TreeSHAP, LinearSHAP, and linear intrinsic explanations). We evaluate the faithfulness of our explanations on two simulated time-series anomaly detection datasets and show that they significantly outperform existing approaches that are not designed for ROCKET-based models.

[1]This work has been accepted for publication at the 2025 IEEE 21st International Conference on Automation Science and Engineering (CASE2025).

- We apply our approach to a real-world manufacturing anomaly prediction dataset and demonstrate our approach's ability to provide effective explanations.

2 Related Work

2.1 Time Series Classification

A multivariate time series consists of multiple sequences of data points indexed in time order. Formally, it can be represented as:

$$\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \ldots, (\mathbf{x}_n, t_n)\}$$

where $\mathbf{x}_t = (x_t^1, x_t^2, \ldots, x_t^d)$ represents a d-dimensional vector at each time step t, with each component x_t^i corresponding to a different variable or sensor reading.

Time series data are prevalent in fields like finance [29], manufacturing [40], and climatology [16], often characterized by trends, seasonality, and outliers. Trends capture long-term changes, while seasonality reflects repeating cycles at fixed frequencies.

Unlike Time Series Forecasting, which aims to predict continuous future sequences, Time Series Classification (TSC) [42] focuses on assigning discrete categorical labels to an entire time series based on its temporal patterns and features.

Formally, a TSC model is defined as a function:

$$f : \mathcal{S} \to \mathcal{Y}$$

where $\mathcal{S}$ represents the space of time series, and $\mathcal{Y}$ is the set of possible class labels. The goal of TSC is to learn a classification model from a dataset

$$\mathcal{D} = \{(s_1, y_1), \ldots, (s_n, y_n)\}$$

where s_i is a time series and y_i is its corresponding categorical target label. Conceptually, TSC can be understood as the process of mapping temporal patterns to specific target values.

TSC encompasses various algorithms, with methods based on convolutional kernels achieving state-of-the-art performance [42]. Among them, methods based on random convolutional kernels (e.g., ROCKET [13], MINIROCKET [14], MultiROCKET [43]) efficiently extract features using a large number of randomly initialized kernels, while deep learning-based models (e.g., InceptionTime [26], ResNet [24]) utilize trained convolutional kernels to capture hierarchical temporal dependencies.

2.2 Random Convolutional Kernel Transformation

Random Convolutional Kernel Transform (ROCKET), proposed by Dempster et al. [13], utilizes a large number of randomly initialized convolutional kernels instead of trained ones. They demonstrated that ROCKET excels in computational efficiency and shorter training time compared to trained convolutional layers by extracting discriminative features without requiring extensive parameter tuning. Additionally, methods based on trained convolutional kernels often struggle to learn effective kernels on small datasets, whereas random convolutional kernels offer a distinct advantage in such scenarios [11].

A ROCKET-based TSC model comprises two key components: the **Random Convolutional Kernel Transformation** and the **classification model**. In multivariate TSC, the Random Convolutional Kernel Transformation effectively captures temporal patterns and inter-variable relationships, while the classification model uses the latent representations generated by this transformation to deliver accurate predictions.

2.2.1 Random Convolutional Kernel Transformation.

The Random Convolutional Kernel Transformation applies a series of 2D convolutional kernels to the input matrix $\mathbf{X} \in \mathbb{R}^{T \times |V|}$, where T represents the time-series length and $|V|$ denotes the number of variables. Each convolutional kernel is randomly initialized with attributes such as kernel size, kernel weights, bias, padding, and dilation. Instead of operating on the entire set of variables, each kernel independently selects a random subset $V_i \subseteq V$ for transformation, ensuring diverse feature extraction across different subsets of variables.

Each 2D random convolutional kernel, with kernel length L_i and kernel weight $\mathbf{w}_i$, processes an input with a convolutional channel size of $|V_i|$ (corresponding to the number of selected variables) and produces a single output channel. The stride is set to 1 across all convolutions. For simplicity, the following formulations assume a padding size of 0 and a dilation rate of 1. Mathematically, the operation for the i-th random convolution at the j-th output position is defined as:

$$\mathrm{Conv}_i(\mathbf{X})[j] = \mathrm{bias}_i + \sum_{l=0}^{L_i-1} \sum_{v \in V_i} \mathbf{w}_i[l, v] \cdot \mathbf{X}[j + l, v], \qquad (1)$$

The output from each convolutional operation varies in length due to differences in kernel size, padding, and dilation. To obtain a consistent feature representation, two predefined transformations are applied:

- **Global Max Pooling (GMP)**: Selects the maximum value from the convolutional output, capturing the most dominant feature across the receptive field.
- **Proportion of Positive Values (PPV)**: Computes the fraction of positive values in the convolutional output, providing a statistical summary of activation patterns.

Formally, given a convolution output sequence $\mathbf{z} = [z_1, z_2, \ldots, z_m]$, these transformations are defined as:

$$\mathrm{GMP}(\mathbf{z}) = \max_i z_i, \quad \mathrm{PPV}(\mathbf{z}) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}(z_i > 0), \qquad (2)$$

where m is the length of the convolution output and $\mathbb{1}(\cdot)$ is the indicator function.

The latent feature space is defined as the concatenation of these transformed representations across C random convolutions:

$$\mathbf{h} = [\mathrm{GMP}(\mathbf{z}_1), \mathrm{PPV}(\mathbf{z}_1), \ldots, \mathrm{GMP}(\mathbf{z}_C), \mathrm{PPV}(\mathbf{z}_C)] \qquad (3)$$

where $\mathbf{h} \in \mathbb{R}^{2C}$ serves as the final fixed-length feature representation for downstream classification tasks.

2.2.2 Classification Model.

The classification model leverages the latent feature space to generate final predictions. This model can be any classifier, such as LR or XGBoost. By operating on the transformed feature space, the classifier effectively captures temporal patterns and relationships within the time series, enabling accurate and reliable classification.

2.3 Explainable Machine Learning

While machine learning and deep learning models achieve high accuracy, they often lack explainability, which is crucial in fields like healthcare [10], risk management [9], and manufacturing [3].

Explainability can be categorized into intrinsically interpretable models and post-hoc explanations. The former include linear models, decision trees, rule-based models, and Generalized Additive Models (GAMs) [22], which are transparent by design. However, these models often trade off explainability for performance, necessitating post-hoc explanations.

Post-hoc methods can be model-agnostic or model-specific. Model-agnostic approaches explain predictions without relying on the model's internal structure. Instead, they approximate feature importances by perturbing inputs, analyzing their effect on predictions, or fitting simpler surrogate models to mimic the original model's behavior. A key consideration in post-hoc methods is faithfulness [2], which refers to how accurately the explanation reflects the model's actual reasoning. *Global* methods, such as Partial Dependence Plots (PDP) [19] and Accumulated Local Effects (ALE) [5], summarize how features influence predictions across the entire dataset. In contrast, *Local* methods, like LIME [41] and KernelSHAP [35], explain individual predictions by estimating feature importances for specific instances. Model-specific techniques, such as LinearSHAP [35], TreeSHAP [34], gradient-based methods [7], and LRP, leverage model structures to explain predictions.

Several studies have been applying and improving ROCEKT, but only limited work has focused on its explainability. Naretto et al. [37] use SHAP to explain ROCKET by treating the original input time series as tabular data and computing SHAP-based feature importances for the input features. Additionally, the method proposed by Tikabo and Touray [44] and X-ROCKET [15] can provide information on which convolutional channels are most frequently activated and which dilation rates are most influential in forming the top latent representations. However, these approaches do not offer input-level feature importances.

3 Methodology

The random convolutional kernel transformation enhances predictive performance by capturing temporal dependencies in time-series data. However, this comes at the cost of explainability. The transformation introduces complexity and randomness, making it difficult to trace how the original input contributes to predictions. The latent feature space, generated through randomly initialized convolutions, obscures direct relationships between inputs and outputs, making conventional explanation techniques ineffective.

Since the classification model operates on the transformed representations, applying explanation techniques to the classification model would lead to explanations that are expressed in terms of the latent feature space (e.g., feature importance scores to each of the GMP or PPV features). This limitation prevents attributions of explanations to original time series data. An alternative approach is to apply model-agnostic post-hoc explanation methods, such as KernelSHAP and LIME, directly to the original input. However, these methods do not capture temporal relationships in time-series data. KernelSHAP quantifies feature contributions based on marginal effects, while LIME approximates the model locally using perturbed samples. Both treat features independently and disregard sequential dependencies, making them more suitable for tabular data [12].

To provide accurate explanations for ROCKET-based TSC models, we propose **ROCKET-LRP**, a two-stage post-hoc approach specifically designed for the ROCKET model. This hybrid method combines feature-level explanations derived from either model-agnostic post-hoc techniques or intrinsically interpretable model-based explanations with LRP's capacity to trace and attribute the temporal relationships captured by the convolutional kernels.

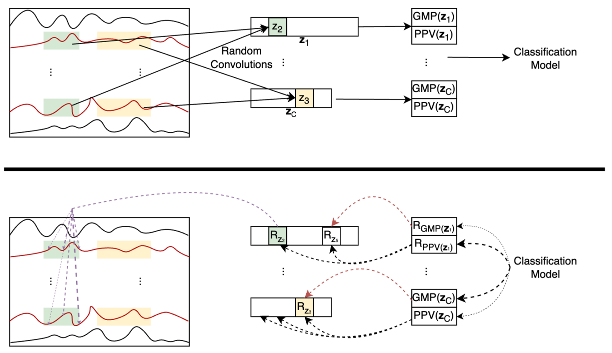

Figure 1: The top diagram shows the structure of the ROCKET-based TSC model. The bottom diagram illustrates the workflow of ROCKET-LRP: The first stage computes local feature importances in the latent space. Feature importances are then propagated to the output of random convolutional kernels following the GMP and PPV rules. Finally, feature importances are propagated to the original inputs using the LRP for convolutions, based on the weighted contributions of inputs to the convolution operation.

3.0.1 *First Stage.*

In the first stage, we focus on computing local explanations for the predictions of the classification model. The explanation methods fall into two categories: those designed for intrinsically interpretable models, which can inherently provide local feature importances, and post-hoc local explanation methods that are applicable to any model.

For intrinsically interpretable models, local explanations are often directly provided by the model itself. For example, in linear models, global feature contributions can be inferred directly from their coefficients, and local feature importances can be expressed as the product of the coefficient and the corresponding input feature value [4]. Similarly, models like GAMs provide built-in local feature importances by design, as they model the target as a sum of functions of individual features [22].

While most machine learning models are not intrinsically interpretable, post-hoc explanation methods provide a powerful tool to compute local feature importances. KernelSHAP and LIME are model-agnostic methods, i.e., they can be applied regardless of the underlying model structure. Although KernelSHAP is computationally inefficient, if the model type is known (e.g., linear or tree-based), model-specific SHAP methods can be used. For example, LinearSHAP is tailored for linear models, and TreeSHAP is designed for tree-based models. These methods are significantly more efficient while maintaining accuracy in their explanations.

Due to the large number of latent features, a filtering step is applied. The filter sets the lowest $\alpha\%$ of the computed feature importances to zero to reduce noise in subsequent computations.

3.0.2 Second Stage. In the second stage, the local feature importances of the latent features are propagated to the original input using Layer-wise Relevance Propagation [6], which was originally developed for distributing relevance scores through the layers of a neural network. We utilize this technique for the first time to propagate local feature importances through ROCKET-based models to the original inputs.

We use $R_{\mathbf{h}}$ to denote the feature importances for the latent features computed based on the classifier in the first stage. $R_{\mathbf{h}}$ consists of two types of feature importance values corresponding to the two pooling operations: GMP and PPV. Formally, it is defined as $R_{\mathbf{h}} = \{R_{\mathrm{GMP}(\mathbf{z}_1)}, R_{\mathrm{PPV}(\mathbf{z}_1)}, \dots, R_{\mathrm{PPV}(\mathbf{z}_C)}\}$. In addition, let $R_{\mathbf{Z}} = \{R_{\mathbf{z}_1}, R_{\mathbf{z}_2}, \dots, R_{\mathbf{z}_C}\}$ denote the local feature importances assigned to the outputs of the random convolutions before pooling, where each $R_{\mathbf{z}_i}$ corresponds to the local feature importances attributed to the output of the i-th random convolution. $R_{\mathbf{z}_i}[j]$ denotes the importance assigned to the j-th position of $R_{\mathbf{z}_i}$. Each feature importance in $R_{\mathbf{Z}}$ consists of contributions propagated from the importances of GMP and PPV features in $R_{\mathbf{h}}$:

$$R_{\mathbf{z}_i}[j] = R_{\mathbf{z}_i}^{\mathrm{GMP}}[j] + R_{\mathbf{z}_i}^{\mathrm{PPV}}[j], \tag{4}$$

The propagation of these importance values to the corresponding convolution outputs in $R_{\mathbf{Z}}$ is defined as follows:

- GMP: If the local feature importance $R_{\mathbf{h}_i}$ in the latent representation is derived using GMP, the entire importance is assigned to the maximum value in the output of the random convolution:

$$R_{\mathbf{z}_i}^{\mathrm{GMP}}[j] = \begin{cases} R_{\mathrm{GMP}(\mathbf{z}_i)}, & \text{if } j = \arg\max \mathbf{z}_i, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

  where $\arg\max \mathbf{z}_i$ identifies the position of the maximum value in the convolution output.
- PPV: If the feature importance $R_{\mathbf{h}_i}$ is derived using PPV, the importance is equally distributed among all positive values in the output of the random convolution

$$R_{\mathbf{z}_i}^{\mathrm{PPV}}[j] = \begin{cases} \frac{R_{\mathrm{PPV}(\mathbf{z}_i)}}{N_i^+}, & \text{if } \mathbf{z}_i[j] > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

  where

$$N_i^+ = \sum_k \mathbb{1}(\mathbf{z}_i[k] > 0)$$

  represents the total number of positive values in the convolution output.

After obtaining the local feature importances for the outputs of the convolutions, these values need to be propagated to the raw input through the random convolutional kernels. The convolution operation is analogous to the linear layer in LRP, and the formulation for the propagation [6] can be expressed as follows:

$$R_{\mathbf{X}}[j,v]_i = \sum_k \frac{\mathbf{X}[j,v] \cdot \mathbf{w}_i[j-k,v] \cdot R_{\mathbf{z}_i}[k]}{\sum_{l=0}^{L_i-1} \sum_{v_i \in V_i} \mathbf{X}[k+l,v] \cdot \mathbf{w}_i[l,v] + \epsilon}, \tag{7}$$

where:

- $R_{\mathbf{X}}[j,v]_i$ represents the local feature importance assigned to the j-th time step and v-th variable in the entire raw input from the i-th random convolution.
- k is constrained by

$$\max(0, j-L+1) \le k \le \min(j, T-L)$$

- ϵ is a small constant that manages the distribution of feature importance when the product of activation and weight is small. As ϵ increases, inputs with smaller contributions receive feature importance closer to zero.

By aggregating the local feature importances from all random convolutions, the local feature importances for the raw input are obtained as:

$$R_{\mathbf{X}}[j,v] = \sum_i R_{\mathbf{X}}[j,v]_i, \tag{8}$$

4 Experiments on Simulated Datasets

To evaluate the faithfulness of our method's explanations, we conducted experiments on two synthetic datasets: Shapelet Data and Spike Data [45]. The generation process for these datasets is detailed below. Each dataset includes 2,000 samples, with 80% for training and 20% for testing. We conducted experiments using two ROCKET-based TSC models, each using a different classifier: ROCKET with XGBoost (ROCKET-XGBoost) and ROCKET with LR (ROCKET-LR). ROCKET-XGBoost was trained using default hyperparameters, while ROCKET-LR employed an l_1 penalty with $C = 0.01$ to encourage sparse coefficients. We observed that both models achieved a testing accuracy of at least 95%, ensuring the models effectively learned the patterns within the data. We then compared the faithfulness of explanations generated by ROCKET-LRP with explanations generated by KernelSHAP, Permutation, and LIME applied directly to the entire ROCKET-based TSC model, following the approach used by Naretto et al. [37]. However, because these methods cannot handle temporally structured data, we flattened the input before applying them. This process did not affect the ROCKET model itself, and the outputs remained unchanged. In ROCKET-LRP, the filtering threshold α is set to 30 for all experiments.

The objective in both datasets was to detect synthetic shapelets or anomalies injected into the time series, with a label of 1 indicating their presence. In our evaluation, we applied the explanation methods to the testing dataset, focusing only on samples that were correctly classified by the model and had a target label of 1, analyzing the key features contributing to the detected anomalies or shapelets. When applying ROCKET-LRP to the ROCKET-XGBoost model, we computed local feature importances using TreeSHAP. For the ROCKET-LR model, we computed local feature importances using either LinearSHAP or the model's coefficients and reported the results for each of the approaches.

4.1 Shapelet Data

We consider the task of identifying shapelets embedded within time series data. A shapelet is defined as a sine wave of length 10, which may be present at a random location within the time series with added noise (see Appendix A for details on the experimental setting). The classification task determines whether a shapelet is present, and the explanation methods are utilized to determine the top 10 most relevant features, i.e., time steps in the time series, contributing to the model's decision.

To assess the faithfulness of ROCKET-LRP, we compute the overlap between the top 10 features determined by each method and the actual ground truth features corresponding to the shapelet. We report mean and percentile overlaps (p90, p50, p10). Additionally, we measure the number of cases where an explanation is completely incorrect (#0), i.e., it does not overlap with any of the shapelet features.

We conduct experiments on two datasets with time series lengths of 50 and 200, each containing a shapelet of length 10 embedded at a random location in some instances. We compare ROCKET-LRP against KernelSHAP, Permutation Feature Importance, and LIME across two classifiers: ROCKET-LR and ROCKET-XGBoost. Table 1 shows the results for the ROCKET with the XGBoost model. Table 2 shows the results for the ROCKET with the LR model.

Table 1: Explanation Overlap with Ground-Truth Features on ROCKET-XGBoost

Method	Mean	p90	p50	p10	#0
Time Series Length = 50					
ROCKET-LRP(TreeSHAP)	**8.65**	**9**	**8**	**7**	0
KernelSHAP	6.38	8	6	5	0
Permutation	6.14	7	6	5	0
LIME	1.88	3	2	1	16
Time Series Length = 200					
ROCKET-LRP(TreeSHAP)	**8.25**	**9**	**8**	**7**	0
KernelSHAP	6.15	7	6	5	0
Permutation	6.20	7	6	5	0
LIME	0.41	1	0	0	183

We observe that ROCKET-LRP consistently achieves the highest mean overlap across both classifiers (ROCKET-LR and ROCKET-XGBoost) and time series lengths (50 and 200), outperforming KernelSHAP, Permutation, and LIME. ROCKET-LRP (Coefficient) significantly outperforms ROCKET-LRP (LinearSHAP) in explaining the ROCKET-LR model; however, ROCKET-LRP (LinearSHAP) still obtains a higher mean (and median) overlap with the ground truth compared to the baselines.

4.2 Spike Data

We consider the task of detecting whether a spike is present in a given relevant feature. Each sample consists of three independent sequences, all with a length of k. The sequences are generated using the TimeSynth package [36] and are based on a second-order Nonlinear Autoregressive Moving Average(NARMA), which models nonlinear dependencies on previous values and noise. For each sequence in a sample, there is a probability of adding random spikes,

Table 2: Explanation Overlap with Ground-Truth Features on ROCKET-LR

Method	Mean	p90	p50	p10	#0
Time Series Length = 50					
ROCKET-LRP (LinearSHAP)	7.53	**9**	**8**	4	0
ROCKET-LRP (Coefficient)	**8.39**	**9**	**8**	**7**	0
KernelSHAP	6.38	8	6	5	0
Permutation	6.14	7	6	5	0
LIME	1.88	3	2	1	24
Time Series Length = 200					
ROCKET-LRP (LinearSHAP)	6.84	**9**	7	2	5
ROCKET-LRP (Coefficient)	**8.21**	**9**	**8**	**7**	0
KernelSHAP	6.22	7	6	5	0
Permutation	6.31	8	6	5	0
LIME	0.50	1	0	0	168

and the ground truth label indicates whether the first sequence contains any spike ($y = 1$) or not ($y = 0$). The indices of spikes in the first sequence are recorded (see Appendix A for details on the experimental setting).

We assess the faithfulness of explanations based on accuracy. Specifically, we consider an explanation to be correct if the feature with the highest local importance falls within an index range of ± 1 from an actual spike, as identifying a spike requires comparing its value to its neighboring data points. We evaluate ROCKET-XGBoost and ROCKET-LR on time series of lengths 30 and 60, comparing the accuracy of different explanation methods.

Table 3: Explanation Accuracy for ROCKET-XGBoost on Spike

Method	Length 30	Length 60
ROCKET-LRP	**98.06**	**94.80**
KernelSHAP	65.12	38.66
Permutation	83.72	86.52
LIME	6.59	4.09

Table 4: Explanation Accuracy for ROCKET-LR on Spike

Method	Length 30	Length 60
ROCKET-LRP (SHAP)	**99.22**	**98.48**
ROCKET-LRP (Coefficient)	**100.00**	**98.48**
KernelSHAP	84.71	77.65
Permutation	90.59	80.68
LIME	8.63	10.61

The results in Table 3 and Table 4 show that ROCKET-LRP performed significantly better than KernelSHAP, Permutation, and LIME for both ROCKET-XGBoost and ROCKET-LR. In this experiment, ROCKET-LRP (LinearSHAP) and ROCKET-LRP (Coefficient) achieved comparable performance, with accuracies in the range of 98% to 100%. These results indicate that ROCKET-LRP effectively explains the behavior of the ROCKET-based TSC model.

5 Experiments on Real Manufacturing Dataset

The datasets used in this study come from a manufacturing facility. The manufacturing process undergoes multiple processing stages before inspection. Sensors placed along the production lines record parameters such as temperature and pressure, while defects, including scratches and foreign objects, are documented during the inspection. As the production is continuous and lacks unique identifiers to directly link sensor readings to specific anomaly values, the model needs to learn to correlate anomaly inspections with earlier sensor readings from the time series data.

The dataset spans two years and consists of four components: sensor data, inspection data, stability data, and sensor information. Sensor data captures readings from various sensors at 10-second intervals. Inspection data records detected anomalies at corresponding timestamps. Stability data tracks system and product stability, with only periods where both are stable being used, resulting in some data gaps. Sensor information included unit specifications and time corrections, which represented the expected time gap between the time a product passes a sensor and the time it reaches the inspection.

In this experiment, we developed a model that predicts the number of anomalies in the next hour's inspections based on sensor data from the previous 12 hours, aggregated into one-hour bins by using the mean.

5.1 Data Preparation

To mitigate the impact of highly correlated sensor values, we applied Pearson Correlation Filtering to the sensor data. Pearson correlation [39] quantifies the linear relationship between two variables. Let the full set of sensor data be defined as $X_{\text{full}} = \{X_1, X_2, \ldots, X_n\}$, where each X_i represents a time-series sensor reading from sensor i. After standardizing the training data, we computed pairwise correlations and identified the most highly correlated sensor pair. The sensor closest to the end of the pipeline (based on time correction) was removed, as downstream sensors are more likely to be influenced by upstream ones. This process was repeated iteratively until the highest correlation between any remaining sensor pair fell below a threshold of 0.65. The resulting filtered sensor set was used for subsequent analysis.

Due to the temporal nature of the data, we performed a monthly train-validation-test split [21]. Each 30-day period was treated as a unit, with the first 55% used for training, the next 15% for validation, and the remaining 30% for testing.

After data splitting, the sensor and inspection data were aligned based on timestamps, and unstable hours were removed. To ensure temporal continuity, a rolling window approach with a step size of 1 was applied, generating 13-hour segments for the training, validation, and testing sets. In each segment, the first 12 hours of sensor data were used as input, while the anomaly count in the final hour served as the target.

5.2 Quantile-based Classification

The accurate prediction of anomaly counts is challenging. In manufacturing, higher anomaly counts require greater attention, and factory managers are typically more concerned with anomaly ranges rather than exact counts. We, therefore, opt to categorize anomalies into four quantiles based on the distribution in the training set [38]. The goal is to use the 12-hour historical sensor data to predict the probability of the next hour's anomaly count falling into a specific quantile. Mathematically, this can be represented as:

$$f(\mathbf{X}) = \mathbb{P}(Q_i|\mathbf{X}), i \in \{1, 2, 3, 4\}$$

where f is the model, and $\mathbf{X} \in \mathbb{R}^{35\times 12}$ represents the input 12-hour data from 35 sensors.

5.3 Experiment Setting

Hyperparameter tuning is conducted using GridSearch [30]. Due to the stochastic nature of the random convolutional kernel transformation, each model is trained three times for three random seeds, and we report the average performance for evaluation. For simplicity, hyperparameter tuning is performed only on the first seed, and the selected parameters are fixed for the subsequent runs.

For models based on the LR classifier, we applied standardization scaling to the input before feeding it into the classifier [25]. For the LR baseline, this scaling is applied to the original data, while for ROCKET-LR, it is applied to the latent space features entering the classifier. Notably, standardization preserves the relative importance of features.

In our experiments, the number of random convolutional kernels is set to 10,000, with kernel sizes randomly selected from {5, 7, 9}.

5.4 Results

Due to the temporal nature of our dataset, a classifier that can effectively utilize temporal patterns is crucial. We first evaluate ROCKET by comparing its performance against LR and XGBoost. Accuracy (ACC) and Mean Absolute Error (MAE) were used for evaluation, with MAE chosen due to the ordinal nature of our quantile-based class definitions, where adjacent classes are considered more similar [20]. MAE, therefore, ensures that misclassifications involving distant classes are penalized more heavily than those involving closer classes.

Table 5: Performance comparison of different models based on Accuracy and Mean Absolute Error

Model	ACC ↑	MAE ↓
ROCKET-XGBoost	**0.68**	**0.36**
ROCKET-LR	0.61	0.46
XGBoost	0.63	0.44
LR	0.55	0.50

The results in Table 5 demonstrate that incorporating the random convolutional kernel transformation significantly enhances the performance of both XGBoost and LR, indicated by both higher accuracy and lower MAE. Next, we applied ROCKET-LRP to ROCKET-XGBoost to determine the feature importances of the inputs. While ROCKET-LRP can also be applied to ROCKET-LR, we focus on ROCKET-XGBoost due to its superior performance on this dataset. As in previous experiments, α is set to 30. Since the manufacturing dataset lacks ground truth for evaluating explanation accuracy, we instead selected the top features determined by ROCKET-LRP and masked the remaining features before retraining ROCKET-XGBoost on the masked data. Our hypothesis is that if the determined features are truly important, a model trained on them should maintain significant discriminative information and achieve good performance [31].

In our experiment, we computed the local feature importances for each training data instance and retained only the top $k\%$ features while setting all other features to zero. We then retrained the ROCKET-XGBoost model on the masked dataset. This approach allowed us to assess whether the top $k\%$ of features contain the discriminative information necessary for accurate predictions. The results were compared against those obtained using KernelSHAP, Permutation, and LIME, as well as a baseline where $k\%$ of features were randomly selected during training.

Table 6: Performance Comparison of Explanation Methods Based on Accuracy and MAE

Accuracy (ACC) ↑			
Method	75%	50%	25%
ROCKET-LRP	**0.66**	**0.62**	**0.52**
KernelSHAP	0.63	0.48	0.33
Permutation	0.59	0.51	0.28
LIME	0.60	0.38	0.20
Random	0.48	0.40	0.36
Mean Absolute Error (MAE) ↓			
Method	75%	50%	25%
ROCKET-LRP	**0.40**	**0.48**	**0.52**
KernelSHAP	0.44	0.81	1.18
Permutation	0.48	0.68	1.37
LIME	0.46	1.07	1.17
Random	0.65	1.11	1.16

The results in Table 6 demonstrate that using ROCKET-LRP for feature masking consistently outperforms the baselines. We observe that ROCKET-LRP effectively preserves critical features, indicated by significantly smaller degradation in performance compared to other baselines. In particular, when retaining only 25% of the features, all explainable baselines perform even worse than the random baseline, except for ROCKET-LRP.

6 Conclusion

We propose a post-hoc explanation method for the ROCKET model, designed to capture and analyze the temporal relationships inherent in the data. Our approach integrates LRP with either model-agnostic post-hoc or intrinsic local explanation techniques and propagates the local explanations computed for the latent features in ROCKET back to the original time series data. By leveraging ROCKET-LRP, we aim to bridge the gap between the strong performance of random convolution-based models and explainability. Our experiments across two synthetic datasets and a real-world manufacturing dataset show that ROCKET-LRP effectively identifies important features. Our work raises interesting directions for future work, including extending ROCKET-LRP to Time Series Extrinsic Regression tasks and exploring additional predictive models and local explanation techniques.

References

[1] Zeina Aboulhosn, Ahmad Musamih, Khaled Salah, Raja Jayaraman, Mohammed Omar, and Zeyar Aung. 2024. Detection of manufacturing defects in steel using deep learning with explainable artificial intelligence. *IEEE Access* (2024).

[2] Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. 2024. Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models. *arXiv preprint arXiv:2402.04614* (2024).

[3] Imran Ahmed, Gwanggil Jeon, and Francesco Piccialli. 2022. From artificial intelligence to explainable artificial intelligence in industry 4.0: a survey on what, how, and where. *IEEE Transactions on Industrial Informatics* 18, 8 (2022), 5031–5042.

[4] Adewale Akinfaderin, Matthew Chasse, Michele Donini, and Benjamin Fenker. 2022. Machine Learning Model Interpretability with AWS. AWS Prescriptive Guidance documentation. https://docs.aws.amazon.com/prescriptive-guidance/latest/ml-model-interpretability/overview.html Amazon Web Services, updated February 28, 2022.

[5] Daniel W Apley and Jingyu Zhu. 2020. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82, 4 (2020), 1059–1086.

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.

[7] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *The Journal of Machine Learning Research* 11 (2010), 1803–1831.

[8] Ivanna Baturynska, Oleksandr Semeniuta, and Kristian Martinsen. 2018. Optimization of process parameters for powder bed fusion additive manufacturing by combination of machine learning and finite element method: A conceptual framework. *Procedia Cirp* 67 (2018), 227–232.

[9] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. 2021. Explainable machine learning in credit risk management. *Computational Economics* 57, 1 (2021), 203–216.

[10] Hongbo Chen, Eldan Cohen, Dulaney Wilson, Myrtede Alfred, et al. 2024. A Machine Learning Approach with Human-AI Collaboration for Automated Classification of Patient Safety Event Reports: Algorithm Development and Validation Study. *JMIR Human Factors* 11, 1 (2024), e53378.

[11] Wen Xin Cheng, Ruobin Gao, PN Suganthan, and Kum Fai Yuen. 2022. EEG-based emotion recognition using random Convolutional Neural Networks. *Engineering applications of artificial intelligence* 116 (2022), 105349.

[12] Jonathan Crabbé and Mihaela Van Der Schaar. 2021. Explaining time series predictions with dynamic masks. In *International Conference on Machine Learning*. PMLR, 2166–2177.

[13] Angus Dempster, François Petitjean, and Geoffrey I Webb. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* 34, 5 (2020), 1454–1495.

[14] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. 2021. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 248–257.

[15] dida. 2023. Inside X-ROCKET: Explaining the explainable ROCKET. https://dida.do/blog/inside-x-rocket. Accessed: 2025-05-30.

[16] Claude Duchon and Robert Hale. 2012. *Time series analysis in meteorology and climatology: an introduction*. John Wiley & Sons.

[17] Carlos A Escobar and Ruben Morales-Menendez. 2018. Machine learning techniques for quality control in high conformance manufacturing environment. *Advances in Mechanical Engineering* 10, 2 (2018), 1687814018755519.

[18] Mojtaba A Farahani, MR McCormick, Ramy Harik, and Thorsten Wuest. 2025. Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms. *Robotics and Computer-Integrated Manufacturing* 91 (2025), 102839.

[19] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[20] Lisa Gaudette and Nathalie Japkowicz. 2009. Evaluation methods for ordinal classification. In *Canadian conference on artificial intelligence*. Springer, 207–210.

[21] NB Harikrishnan, R Vinayakumar, KP Soman, and Prabaharan Poornachandran. 2019. Time split based pre-processing with a data-driven approach for malicious url detection. *Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments* (2019), 43–65.

[22] Trevor Hastie and Robert Tibshirani. 1990. *Generalized Additive Models*. CRC Press.

[23] Junyi He and Feng He. 2024. Optimization of Internet Platform Supply Chain Management based on Machine Learning Algorithm. In *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*. IEEE, 1–5.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer*

vision and pattern recognition. 770–778.

[25] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*. John Wiley & Sons.

[26] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[27] Aisulu Ismailova, Zhanar Beldeubayeva, Kuanysh Kadirkulov, Zhanagul Doumcharieva, Assem Konyrkhanova, Dinara Ussipbekova, Ainura Aripbayeva, and Dariga Yesmukhanova. 2024. Forecasting stock market prices using deep learning methods. *International Journal of Electrical and Computer Engineering (IJECE)* 14, 5 (2024), 5601–5611.

[28] Faria Khandaker, Arik Senderovich, Junda Zhao, Eldan Cohen, Eric Yu, Sebastian Carbajales, and Allen Chan. 2024. Transformer models for mining intents and predicting activities from emails in knowledge-intensive processes. *Engineering Applications of Artificial Intelligence* 128 (2024), 107450.

[29] Young Shin Kim, Svetlozar T Rachev, Michele Leonardo Bianchi, Ivan Mitov, and Frank J Fabozzi. 2011. Time series analysis for financial market meltdowns. *Journal of Banking & Finance* 35, 8 (2011), 1879–1891.

[30] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. 2004. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research* 23, 7-8 (2004), 673–692.

[31] Alexandre R Lemos, Thomas W Rauber, and Celso J Munaro. 2023. Feature Selection for Fault Detection in Industrial Processes Based on the SHAP Algorithm. In *2023 15th IEEE International Conference on Industry Applications (INDUSCON)*. IEEE, 1300–1305.

[32] Benjamin Lindemann, Nasser Jazdi, and Michael Weyrich. 2020. Anomaly detection and prediction in discrete manufacturing based on cooperative LSTM networks. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 1003–1010.

[33] Mengyun Liu and Krishnendu Chakrabarty. 2021. Adaptive methods for machine learning-based testing of integrated circuits and boards. In *2021 IEEE International Test Conference (ITC)*. IEEE, 153–162.

[34] SM Lundberg, GG Erion, and SI Lee. 2018. Consistent individualized feature attribution for tree ensembles. arXiv 2018. *arXiv preprint arXiv:1802.03888* 10 (2018).

[35] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[36] J. R. Maat, A. Malali, and P. Protopapas. 2017. TimeSynth: A Multipurpose Library for Synthetic Time Series in Python. [Online]. Available: http://github.com/TimeSynth/TimeSynth.

[37] Francesca Naretto, Roberto Pellungrini, Salvatore Rinzivillo, and Daniele Fadda. 2023. Exphlot: Explainable privacy assessment for human location trajectories. In *International Conference on Discovery Science*. Springer, 325–340.

[38] Guillermo Navas-Palencia. 2020. Optimal binning: mathematical programming formulation. *arXiv preprint arXiv:2001.08025* (2020).

[39] Karl Pearson. 1895. Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London* 58 (1895), 240–242. doi:10.1098/rspl.1895.0041

[40] Robert W Resek. 1966. Investment by manufacturing firms: A quarterly time series analysis of industry data. *The Review of Economics and Statistics* (1966), 322–333.

[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[42] Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I Webb. 2021. Time series extrinsic regression: Predicting numeric values from time series data. *Data Mining and Knowledge Discovery* 35, 3 (2021), 1032–1060.

[43] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I Webb. 2022. MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery* 36, 5 (2022), 1623–1646.

[44] Kamal Tikabo and Pamodou Touray. 2024. Bridging the Gap: Enhancing Explainability in ROCKET.

[45] Sana Tonekaboni, Shalmali Joshi, Kieran Campbell, David K Duvenaud, and Anna Goldenberg. 2020. What went wrong and when? Instance-wise feature importance for time-series black-box models. *Advances in Neural Information Processing Systems* 33 (2020), 799–809.

[46] Sarthak Manas Tripathy, Ashish Chouhan, Marcel Dix, Arzam Kotriwala, Benjamin Klöpper, and Ajinkya Prabhune. 2022. Explaining Anomalies in Industrial Multivariate Time-series Data with the help of eXplainable AI. In *2022 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 226–233.

[47] Abdul Wahid, John G Breslin, and Muhammad Ali Intizar. 2022. Prediction of machine failure in industry 4.0: a hybrid CNN-LSTM framework. *Applied Sciences* 12, 9 (2022), 4221.

[48] Kyung Keun Yun, Sang Won Yoon, and Daehan Won. 2023. Interpretable stock price forecasting model using genetic algorithm-machine learning regressions and best feature subset selection. *Expert Systems with Applications* 213 (2023), 118803.

A Experimental Setting for Simulated Datasets

Shapelet: A shapelet is embedded within a time series by adding a sinusoidal wave defined as: shapelet = $0.5 \times \sin(\text{linspace}(0, 2\pi, 10))$, at a randomly selected position within the sequence. The shapelet is further perturbed with Gaussian noise of standard deviation 0.05 to introduce variability. Approximately 60% of the samples contain shapelets. The number of random convolutional kernels is set to 5,000, with kernel sizes selected from {7, 9, 11}.

Spike: Adapted from [45], with an 80% probability of each sequence containing a random number of spikes, drawn from a Poisson distribution with a mean of 2. The spike locations are randomly chosen, and each spike increases the corresponding data point by 0.3. The number of random convolutional kernels was set to 5,000, with kernel size selected from {2, 4, 6}.

B Hyperparameters for Manufacturing Experiments

Table 7: Grid Search Tuning Space for Different Models

Hyperparameter	Values	Applicable Models
Max Depth	{6, 8, 10}	ROCKET-XGBoost, XGBoost
Num of Estimators	{200, 300}	ROCKET-XGBoost, XGBoost
Learning Rate (LR)	{0.1, 0.2}	ROCKET-XGBoost, XGBoost
Subsample	{0.8, 1.0}	ROCKET-XGBoost, XGBoost
ColSample	{0.6, 0.8}	ROCKET-XGBoost, XGBoost
C	{1, 0.1, 0.01}	ROCKET-LR, LR