# On the Anomaly Detection in Time Series Data with Kernel PCA Algorithm

Uladzimir Charniauski
University of Connecticut

April 29, 2025

**Abstract**

Kernel Principal Component Analysis (Kernel PCA) is a non-linear extension of PCA. Kernel-based methods better model data produced by nonlinear processes by mapping data on higher-dimensional spaces using kernel functions. This feature enables Kernel PCA to capture the complex, nonlinear relationships inherent in many observed real-world data. The goal of this study is to test and analyze the effectiveness of kernel Principal Component Analysis in detecting anomalous points for the time series data. We perform anomaly detection on select time series datasets with recorded anomalies and compare their classification performance with the regular PCA method. We demonstrate the slight performance advantage of the Kernel PCA method in handling anomaly detection on three real-world datasets: circuit water, fluid leaks, and rotor imbalance. We also perform the cross-validation parameter search on a grid for both of considered methods to investigate how the number of anomalies present in time series impact the optimal values of PCA and Kernel PCA key parameters. Finally, we construct the ROC curves to evaluate the model performance in detecting anomalies in terms of the interplay of true positive and false positive rates.

**Keywords:** Kernel method, Novelty detection, PCA, Time Series

## 1 Introduction

The purpose of anomaly (outlier or novelty) detection method is to identify the anomalous data points within a dataset that deviate significantly from the expected behavior defined by the presence of ordinary background points. This process helps to detect unusual or potentially problematic events that may require additional attention. The challenge in this task is that anomalies, by their nature, are rare and often generated by the underlying processes that differ fundamentally from those that produce the majority of the data (2).

Anomaly detection has vast applications in medicine, fraud detection, network security, quality control, and industrial monitoring(1; 2).

Principal Component Analysis (PCA) (3) is a popular dimensionality reduction method that proves its effectiveness in detecting anomalies hidden in datasets. This method reduces the dimensionality of large data sets by transforming complicated sets of variables into smaller ones while preserving most of the information from the large sets. This systematic approach provides a robust framework for detecting anomalies and extracting meaningful patterns from data across a range of applications. However, in many real-world applications, underlying the anomalies arise from complex, nonlinear patterns. This notion notably reduces PCA's ability to accurately detect anomalies in such nonlinear data (4). This implication spurs the interest in exploring other statistical methods suitable for detecting anomalies by adapting to the complexity of patterns exhibited in the dataset.

Kernel methods offer a greater flexibility in addressing the non-linearity issue by modeling the data in a transformed, non-linear feature space $F$. A kernel function enables calculations of inner products in $F$ without explicitly calculating the mapping. Existing research introduced several popular kernel methods tackling various classification and detection problems. One-Class Support Vector Machines (OC-SVM)(1) is a popular method for anomaly detection. This method separates the data from the origin in $F$, or, for a Gaussian radial-basis-function (rbf) kernel, spherically encloses the data in $F$. Parzen Window(1), or sometimes Kernel Density Estimation (KDE), is another useful method for anomaly detection task. This one estimates the probability density function of a dataset directly from the samples in $F$ through placing kernel functions at each data points (typically rbf) in $F$ that collectively contribute to the density estimates at any given point. Kernel PCA (1) is a method that extends the original PCA by mapping input data to a high-dimensional (potentially infinite-dimensional) $F$ with a kernel function computing inner-products in $F$. This way, Kernel PCA captures complex features that are not accessible to linear PCA, which makes it particularly useful for datasets with nonlinear structures. The prevailing concensus highlights the importance of employing

kernel-based methods for efficient handling of anomaly detection tasks to attain highly precise accuracy.

Limited studies have investigated the application of kernel methods in anomaly detection in time series. Frehner et.al. (5) introduce a novel kernel-based approach anomaly detection approach tailored for time series data that relies exclusively on normal events during training. The authors proposed an application of Kernel Density Estimation (KDE) that addresses the versality of anomalies by analyzing reconstruction errors to access an empirically derived probability distribution for normal events post-reconstruction. Their method employs autoencoders that are capable of learning normal representation without prior anomaly knowledge. Nevertheless, this study specifically concerns anomaly detection methods that utilize autoencoder architectures, which fails to directly encompass the effectiveness of feature transformations using kernels in anomaly detection. The scarcity of existing literature studying the performance of kernel-based methods in novelty detection across time series data has sparked the initiation of the presented study.

The goal of this paper is to analyze the anomaly detection performance of Kernel PCA with PCA on time series data to test the superiority of Kernel PCA method in anomaly detection task, as shown in a similar comparison study by Hoffman(1). In his experiments, Hoffman demonstrated that Kernel PCA dominates all three methods enlisted above in novelty detection by better generalizing the data and proving itself to be more robust against the noise in data. Because this method showed promising capabilities in detecting anomalies on regular datasets, we are interested in analyzing how well Kernel PCA can classify the anomalies in the data with present temporal dynamics. However, although Hoffman also compared Kernel PCA with Parzen Window and OC-SVM methods, we will only focus on comparing Kernel PCA with its linear counterpart due to computational resources constraint.

The remainder of the paper is organized as follows. In Section 2, we consider three time series datasets and describe the important preprocessing steps for time series data. In Section 3, we expand on the methodology and introduce the model training procedures, explaining the motivation behind selected approaches. We later provide extensive

results for our experiments in Section 4 and demonstrate the accuracy metrics achieved by each considered method. Lastly, in Section 5, we summarize the key discoveries of our study and report the efficiency and comparative advantage of each method in detecting anomalies in different cases of time series data.

# 2 Data

The three time series datasets implemented in this project are the following: Circuit Water, Fluid Leaks, and Rotor Imbalance. We downloaded these datasets from the Skoltech Anomaly Benchmark' Github (6), a public time series repository providing time series data for evaluating anomaly detection methods. Each dataset consists of around 1,000 observations about the technical metrics of experimental behaviors corresponding to each dataset, including the anomaly labels. Although they all contain the same features, the number of anomalies present in each dataset are vastly different. On each dataset, we will test the abilities of PCA and Kernel PCA to detect anomalies. Because those are unsupervised methods, we will not include anomaly labels (0 and 1) in the training set. However, for the purpose of our research, we will use anomaly labels in each dataset for parameter search and model evaluation tasks. Table 1 summarizes the variables included in each given dataset.

There are several pre-processing steps that we will take to prepare our data for training. First, because each experimental benchmark corresponding to its dataset was recorded for various time durations ranging in 5-20 minutes, we exclude the date-time column from our working data, since these time periods do not give us any useful seasonality patterns. We also exclude the changepoint feature from our analysis. This variable, while informative, explicitly identifies points of structural change or collective anomalies. Including it in the analysis risks introducing data leakage, where models could implicitly rely on this variable as a shortcut to detect anomalies, rather than learning from the underlying patterns in the data itself.

After preprocessing, we begin splitting our data between training, validation, and

| Variables | Range of Values | Description |
|---|---|---|
| X1: datetime | N/A | Represents dates and times of the moment when the value is written to the database |
| X2: Accelerometer1 - RMS (g) | 0-1 | Vibration acceleration (Amount of g units) |
| X3: Accelerometer2 - RMS (g) | 0-1 | Vibration acceleration (Amount of g units) |
| X4: Current (Ampere) | 1-4 | Shows the amperage on the electric motor |
| X5: Pressure (Bar) | $-1$-1 | Represents the pressure in the loop after the water pump |
| X6: Temperature (C) | 85-87 | Shows the temperature of the engine body |
| X7: Thermocouple (C) | 28-32 | Represents the temperature of the fluid in the circulation loop |
| X8: Voltage (Volt) | 220-240 | Shows the voltage on the electric motor |
| X9: RateRMS (L/min) | 13-56 | Represents the circulation flow rate of the fluid inside the loop |
| X10: changepoint | 0,1 | Shows if the point is a changepoint for collective anomalies (0 or 1) |
| Y: Anomaly: | 0,1 | Shows if the point is anomalous (0 or 1) |

Table 1: Summary and Explanation of the Main Variables

testing sets, for selecting which we provide an experimental motivation in future sections. In this project, we will implement the time series cross-validation for searching for the best model hyperparameters, which we also describe more in future sections, so we only split the data between the combined training-validation and testing sets. We use 8/10 of our data for training and validation and leave split the remaining 2/10 of data for testing.

Because we work with time series data, we should rather chronologically split our data than applying random sampling because we expect the autocorrelation in the data. For this purpose, we first divide our data observations for both independent and label features between those that have anomaly label as 0 and 1 in two separate sub-datasets: normal data and anomaly data, respectively. Only then we split our normal and anomalous datasets for training, validation, and testing sets in the same ratios as provided above. Finally, we concatenate the split normal and anomalous sets for each category to prepare our working datasets for anomaly detection tasks. The training-validation-testing splitting is done with the help of methods from `Pandas` library. The training, validation, and testing sets are represented as `NumPy` objects, for creating which we utilize appropriate methods from this library.

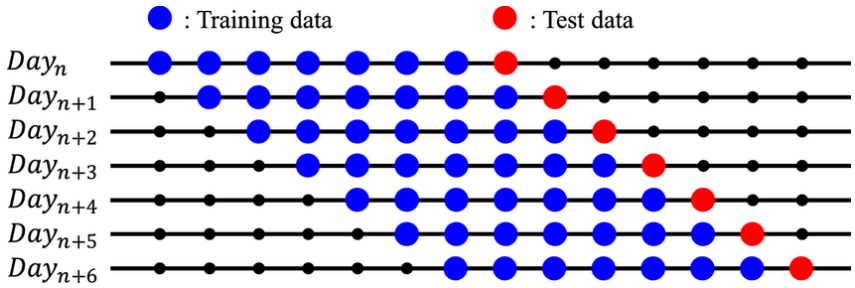


Figure 1: Sliding Window Approach for Time Series Data

We later convert our prepared training, validation, and testing sets into a sliding window format to adjust the experimental conditions to our needs, with the selected default window size of 20 observations. This is a popular size used in practical works with time series data. We also need to scale the data for our training-validation at each cross-validation split and the testing set. For this task, we will employ a standard scaling technique that standardizes our features. For scaling the data, we use `StandardScaler()` method from scikit-learn toolkit from Python.

The sliding window format is an essential tool because it allows models to consider a sequence of past values, which helps in understanding temporal dependencies and detecting deviations from expected patterns. This way, our detection models would first fit on normal sequences and flag windows with high anomaly scores. Indeed, Hoffman did not consider this transformation technique because his experiments did not involve detecting anomalies on time series data. Figure 1 illustrates the mechanism of a sliding window format.

# 3 Methods

In this study, we employ an implementation of PCA and Kernel PCA from the `Scikit-learn` Python library, which provides many essential functions for training and evaluating these models. We will use a Gaussian radial basis function (rbf) for Kernel PCA method, which option is available within the same library. For calculating reconstruction errors and generating AUC (Area-Under-Curve) scores for model evaluation, we extensively use a combination of methods from `Scikit-learn` and `NumPy` libraries. For plotting our scat-

terplots, heatmaps, and drawing ROC (Receiving Operating Characteristics) curves, we use graphing tools from `Matplotlib` Python library.

## 3.1 Application of PCA/Kernel PCA for Anomaly Detection

We provide mathematical details about how PCA/kPAC work in anomaly detection task. We consider $X \in \mathbb{R}^{n \times m}$ to be our given dataset with $n$ samples and $m$ features. The PCA method finds a transformation matrix $W$ that projects the data to the lower-dimensional space through $Z = WX$, where $Z$ is our projected matrix and $W$ contains principal components. The method then reconstructs the data using the inverse transformation $\hat{X} = ZW^T$. The reconstruction errors are then calculated as the squared Euclidean distance between the original and reconstructed data points as $\|X - \hat{X}\|_2^2$, where samples with high reconstruction errors are considered anomalous.

The Kernel PCA further extends the concept of PCA to non-linear relationships by each data point $\mathbf{x}_i \in \mathbb{R}^m$ and mapping it into a higher-dimensional feature space $F$ through $\mathbf{x}_i \rightarrow \Phi(\mathbf{x}_i)$, where $\Phi(\mathbf{x})$ is a vector $\mathbf{x}$ mapped to $F$. The method then performs standard PCA in $F$ on $\Phi(\mathbf{x}_i)$ vectors by using the kernel function $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2\right)$), which replaces the scalar product $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$. Using this product, the method computes the kernel matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2)$ and centers the matrix in $F$ as $K_c = K - \mathbb{1}_m K - K\mathbb{1}_m + \mathbb{1}_m K \mathbb{1}_m$, where $\mathbb{1}_m \in \mathbb{R}^{m \times m}$ is a matrix with all elements equal to $\frac{1}{m}$. Next, the method proceeds with the eigendecomposition by solving $K_c\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$, which yields eigenvalues $\lambda$ and eigenvectors $\boldsymbol{\alpha}$, and normalizing the given eigenvectors $\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_k / \sqrt{\lambda_k}$. After that, the method projects each data point from a sample $\mathbf{x}_i$ onto the principal components through $\mathbf{y}_{ij} = \sum_{k=1}^{n} \boldsymbol{\alpha}_{jk} K_{ik}$. Finally, the method reconstructs each sample in the feature space as $\hat{\mathbf{x}}_i = \sum_{j=1}^{p} \mathbf{y}_{ij} \boldsymbol{\alpha}_j$, where $p$ is the number of retained (largest) principal components, and calculates reconstruction errors as $r(\mathbf{x}_i) = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2$. Again, the samples with high errors indicate anomalies.

## 3.2 Parameter Search

The original Hoffman's evaluation did not include the validation set. He captures the models in the training set, then proceeds with parameter tuning on a testing set. In his case, this was not representative of the real-world implementation of the detection algorithms because anomalous points are unknown during testing. Moreover, since we address the anomaly detection performance on time series data, we enable our models through gridsearch to adapt to specific characteristics and intricate complexities of considered datasets, which will significantly improve the quality of anomaly detection by each method.

We will perform the grid search cross-validation on the validation data for both considered methods to mimic the real-world deployment of these algorithms. In this methodology, our splits maintain chronological order, ensuring training data always precedes validation data. The corresponding training set consists only of observations that occurred prior to the observation that forms the test set. We will still perform the training on a clean portion of data that does not contain anomaly scores, but only the best parameters in the validation set, those that give the highest AUC score from reconstruction errors, are selected for testing. The models are then tested separately on a selected testing set using the best parameter setting discovered. We should also note that because we converted the data to a sliding window format, we lose some of the first data points in the quantity of a window size in our validation and testing sets, so we adjust our anomalies in validation and testing data by the selected window size to compute AUC scores.

The following is a description of a range of grid-seach parameters that we introduce to each algorithm. We consider 37 principal components for both PCA and Kernel PCA models, which equals to the size of the validation set across all datasets. For Kernel PCA, we also consider 20 kernel coefficients (gamma) from 0.01 to 100 along a log scale. This way, the algorithm will check 740 parameters during the validation for each dataset. For each parameter on a grid, we split our combined training and validation data by 5 folds to train our model on a growing training set and validate its performance on a validation set that moves forward with each fold.

# 4 Results

In this section, we first illustrate (Section 4.1) the gridsearch results for our methods for each dataset. After that, we will show (Section 4.2) the ROC curves for each algorithm generated using the best parameter settings.

## 4.1 Parameter Search

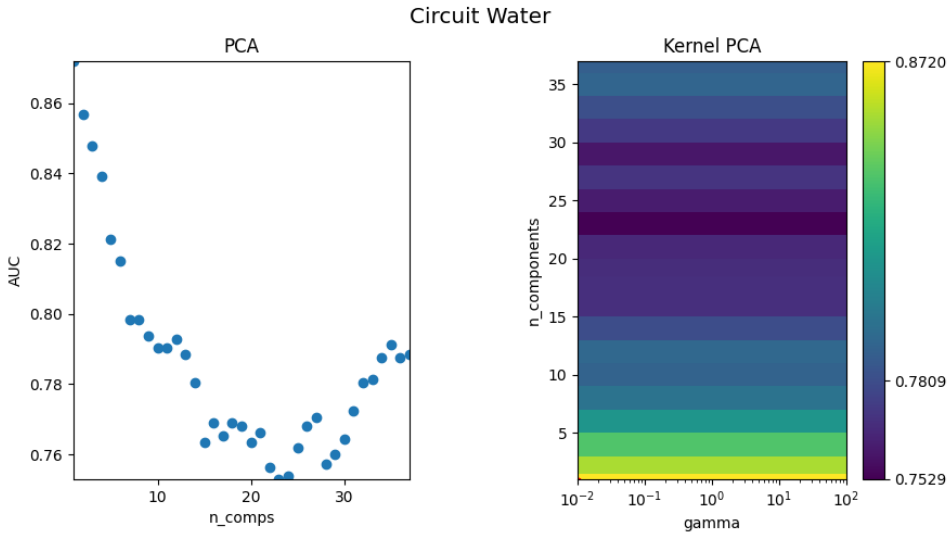


Figure 2: Gridsearch results for the Circiut Water dataset

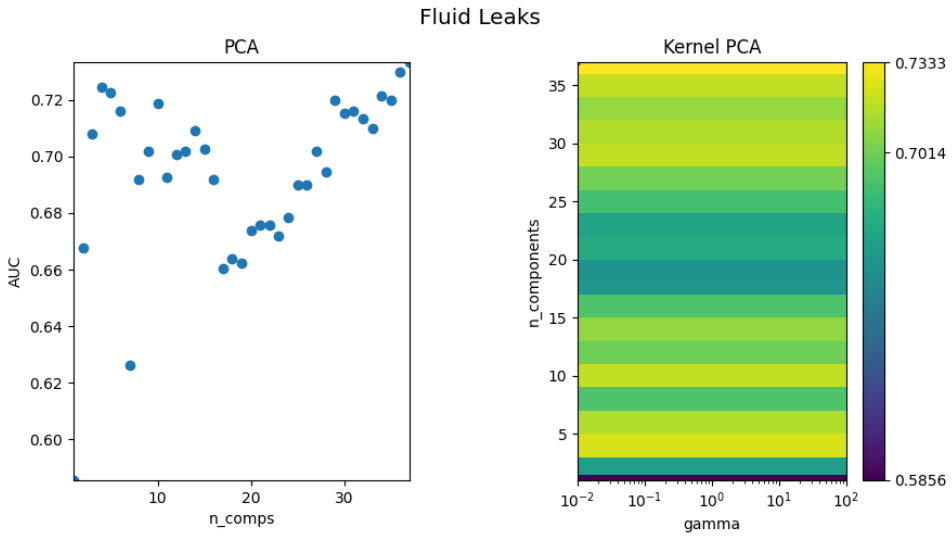


Figure 3: Gridsearch results for the Fluid Leaks dataset

Figures 2-4 show the gridsearch results for PCA and Kernel PCA anomaly detection methods on the validation set for each data set. For PCA, the gridsearch scatterplot

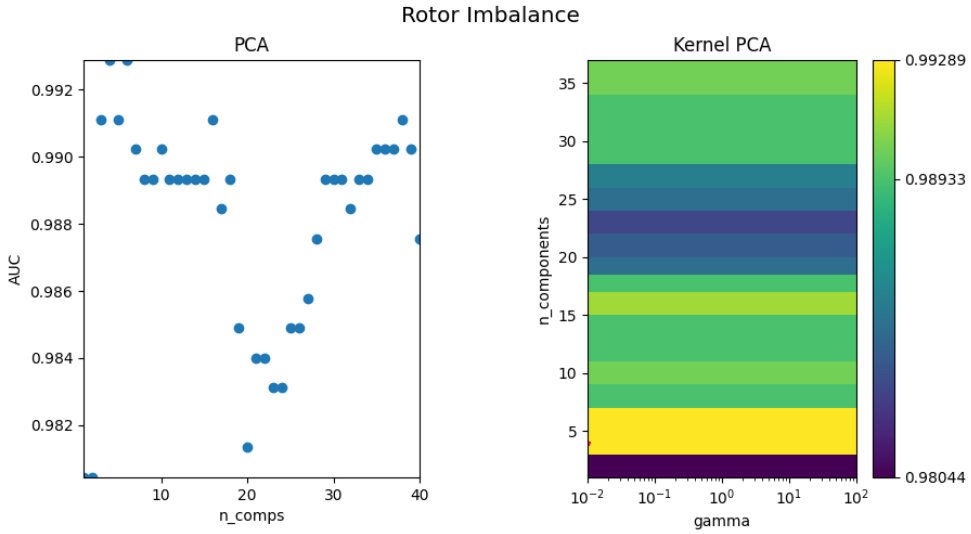


Figure 4: Gridsearch results for the Rotor Imbalance dataset

represents the AUC score for each number of retained principal components. For Kernel PCA, the gridsearch heatmap represents AUC scores for each selection of kernel coefficient gamma and the number of retained principal components in their respective ranges. The red star indicates the best parameter selections in the 2D Kernel PCA grid. The adjacent colorbar represents the range of AUC scores. The central tick mark corresponds to the median AUC over the search. The top and bottom tick marks denote the maximum and minimum respectively.

| Dataset | Anomalies | Method | Best Parameter(s) | Val AUC | Test AUC |
|---|---|---|---|---|---|
| Circuit Water | 586 | PCA | n = 4 | 0.9929 | 0.9594 |
| Circuit Water | 586 | Kernel PCA | gamma=0.01; n = 4 | 0.9929 | 0.9740 |
| Fluid Leaks | 188 | PCA | n = 37 | 0.7333 | 0.6303 |
| Fluid Leaks | 188 | Kernel PCA | gamma=0.01; n = 37 | 0.7333 | 0.6921 |
| Rotor Imbalance | 410 | PCA | n = 1 | 0.8720 | 0.8518 |
| Rotor Imbalance | 410 | Kernel PCA | gamma=0.01; n = 1 | 0.8720 | 0.8913 |

Table 2: Simulation Results Comparing Estimators

Table 2 presents the validation and testing results for both methods on each dataset along with their anomaly characteristics. From the results below, we observe that both methods require the same optimal numbers of principal components. We also see that both PCA and Kernel PCA methods achieve the equal validation AUC scores. This is because the intrinsic dimensionality of our data is governed by linear relationships, so both

methods converge to similar component needs during the validation. For every dataset, the optimal value of a Kernel PCA's gamma parameter is 0.01, which is the lowest value on the entire grid of gammas. The small values of optimal gamma parameters for Kernel PCA indicate that the method globally captures anomalies by introducing smooth, non-linear transformations to the data. On the other hand, those values suggest that the anomalies in our datasets are well captured in a linear space, meaning PCA itself might also be sufficient for effective anomaly detection.

From the table above, we see that Kernel PCA always achieves a higher Test AUC than its linear counterpart. This observation would suggest that while the validation AUC scores may appear similar, Kernel PCA's ability to model complex structures ensures better separation between normal and anomalous data in the test set. This method also produces the testing scores that are less discrepant from the validation scores than in case of PCA. This notion suggests that Kernel PCA generalizes the data better with the achieved optimal components, further highlighting the benefits from using this method for anomaly detection task. It's worth noting that although Hoffman did not use a validation set, our results align with the outcomes of his experiments on anomaly detection, validating his original claim of the effectiveness of a Kernel PCA method in anomaly detection tasks.

## 4.2 The ROC curves

The plots below show the generated ROC curves based on detection on the test set. using the best gridsearch parameters for each detection method. We presented the false positive rate on a log-scale to emphasize the most important region.

The ROC curves in Figure 5 support the findings in Table 2 and illustrate how Kernel PCA generally outperforms PCA across the examined time series datasets. We see that Kernel PCA begins to detect anomalies early on with the optimal parameter setting on Circuit Water dataset, producing higher True Positive (TP) rates than its competing method. On the other hand, in the case of two other datasets, this method starts very low on a TP rate scale and at some point rises up along the TP rate, indicating poor

detection performance, for which the method compensates through random guessing for larger False Positive (FP) rates. This effect is especially detectable in case of Fluid Leaks dataset, where the ROC curves both PCA and Kernel PCA closely imitate the random guessing.

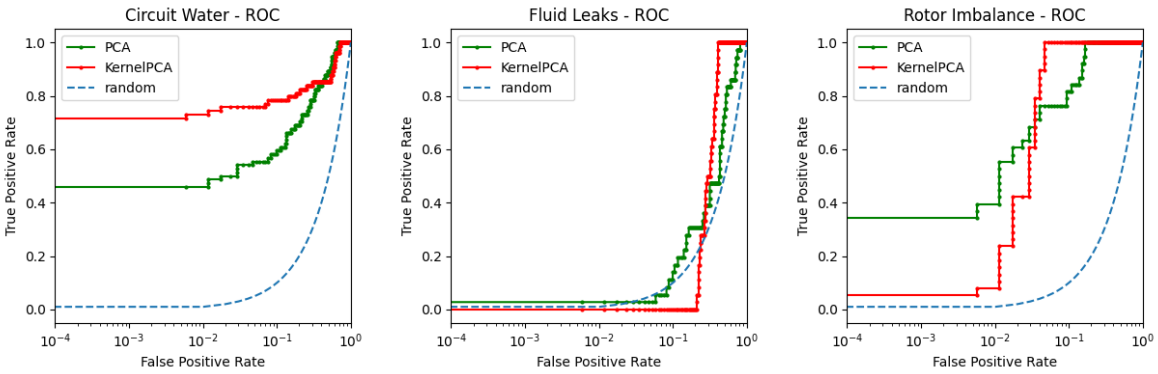


Figure 5: ROC Curves

In contrast, on the Fluid Leaks dataset, both PCA and Kernel PCA perform poorly at low FP rates, with their ROC curves initially resembling random guessing. TP rates for both models increase only at higher FP rates, suggesting that the models can detect anomalies but do so at the cost of mislabeling many normal points. This behavior likely stems from distributional shifts between the training and test sets, which challenge generalization. Nevertheless, Kernel PCA outperforms PCA on the Fluid Leaks datasets by generalizing the data through processing non-linear patterns the latter is unable to handle.

For the Rotor Imbalance dataset, Kernel PCA ultimately achieves a higher AUC score than its linear counterpart. However, PCA produces a smoother ROC curve, whereas Kernel PCA starts nearly at random guessing rates. Kernel PCA, on the other hand, initially performs close to random but eventually surpasses PCA by achieving a higher TP rate as the FP rate increases. These results suggest that Kernel PCA method poorly separates normal points from anomalies initially, but becomes more effective as the decision threshold loosens. Despite this trade-off, Kernel PCA identifies more anomalies overall, highlighting its capacity for nonlinear pattern detection at the cost of higher false alarms.

# 5 Discussion and Conclusion

In this work, we addressed the effectiveness of Kernel PCA in anomaly detection on different time series datasets. This evaluation differs from Hoffman's original work, since we tested our methods on three introduced time series datasets instead of regular baselines utilized in original evaluations. We also employed a time series cross-validation for parameter tuning distinctly performed for each method, which was not the case of considerations for the original study.

We generated insightful results about the PCA and Kernel PCA abilities to detect anomalies in time series across our experiments. Both PCA and Kernel PCA methods selected the same values of principal components for every dataset. This notion indicates that both methods consistently reduce dimensionality by capturing the most informative issues. Such observation also suggests that the underlying structure of each dataset is well-represented in both linear PCA and non-linear Kernel PCA projections, which is evident from near-equal Test AUC scores for each method on every dataset.

The Kernel PCA proved to be effective against PCA in detecting anomalies, which supports Hoffman's claim about the comparative performance of this method. However, because the Test AUC scores for PCA and Kernel PCA are relatively close in value across every experiment, the additional flexibility of Kernel PCA does not significantly improve the representation compared to standard PCA, as the time series data may have a primarily linear structure. That also explains the Kernel PCA selection of small values of gamma as optimal parameters, suggesting that the anomalies are not highly nonlinear, and the sliding window transformation itself has already made them easier to detect using linear relationships.

Our particular comments require the observed AUC scores for validation and testing sets. In each experiment, with the time series cross-validation technique used, both PCA and Kernel PCA achieved the same validation AUC scores. This observation again suggests the linearity in data that allows both methods to converge to the same results. Importantly, since Kernel PCA is prone to overfitting due to its greater flexibility, matching AUC scores suggest that your cross-validation strategy successfully prevented overfitting

and that this non-linear model did not exploit noise as signal. For both methods, the best parameter setting found during the validation set was the best as well during the testing. Thus, although Hoffman did not consider a validation set in his experiments, nor did he tested the anomaly detection capabilities of these methods on a time series data requiring sliding window processing, we are able to successfully reproduce the original anomaly detection experiment after employing the multi-staged preprocessing procedure to our data.

This study investigates the effectiveness of Kernel PCA for anomaly detection in time series data, applied in a setting that deviates substantially from the conditions explored in Hoffman's original work. The results highlight the advantages of Kernel PCA as a novelty detection algorithm, particularly in its ability to capture nonlinear structures in various time series datasets. Future research should further explore the impact of preprocessing strategies—such as window size selection in sliding window transformations—on detection performance, as well as the development of specialized kernel methods tailored to temporal patterns. Although this study did not include comparisons with OC-SVM or Kernel Density Estimation (KDE), a systematic evaluation of these methods alongside PCA and Kernel PCA could provide deeper insights into their relative strengths. Finally, assessing the scalability and efficiency of Kernel PCA on large-scale, high-frequency time series data remains an important direction for establishing its practical utility in real-world anomaly detection scenarios.

# Acknowledgments

The author would like to thank Professor Qian Yang, from the School of Computing, University of Connecticut, who offered insightful feedback and invaluable support on the experimental and theoretical parts, contributing to the success of this study.

# References


[1] Hoffman, H. (2007). Kernel PCA for novelty detection. Pattern Recognition, 40(3), 863–874. https://doi.org/10.1016/j.patcog.2006.07.009

[2] Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenkova, B., Schubert, E., Assent, I., & Houle, M. E. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. Pattern Recognition, 77, 43–57. https://doi.org/10.1016/j.patcog.2016.02.001

[3] Ringberg, H., Soule, A., Rexford, J., & Diot, C. (2007). Sensitivity of PCA for traffic anomaly detection. ACM SIGMETRICS Performance Evaluation Review, 35(1), 109–120. https://doi.org/10.1145/1254882.1254895

[4] Goldstein, M., & Uchida, S. (2016). A comprehensive survey of anomaly detection techniques for high-dimensional big data. Journal of Big Data, 7(42). https://doi.org/10.1186/s40537-020-00320-x

[5] Detecting anomalies in time series using kernel density approaches. (2024). IEEE Transactions on Neural Networks and Learning Systems, 35(1), 123–135. https://doi.org/10.1109/TNNLS.2024.3045678

[6] Waico. (2023). SKAB - Skoltech Anomaly Benchmark [GitHub repository]. GitHub. Retrieved March 29, 2025, from https://github.com/waico/SKAB/tree/master