Multivariate Time Series Anomaly Detection

By- Abhijeet Kundu

Code and Output-
https://github.com/abhijeetkundu8/Multivariate_Time_Series_Anamoly_Detection.git

Problem Background

Performance management systems continuously monitor asset health using data from sensors, IoT devices, and other sources. They analyze this data to identify potential issues, predict failures, and optimize maintenance schedules. The process includes identifying data points, patterns, or events that deviate significantly from what is considered normal or expected within a dataset. This helps organizations transition from reactive to proactive and predictive maintenance strategies, leading to increased efficiency, reduced costs, and improved asset reliability.

Objective

Develop a Python-based machine learning solution to detect anomalies in multivariate time series data and identify the primary contributing features for each anomaly.

PCA-Based Anomaly Detection System

1. Proposed Solution

Detailed Explanation of the Proposed Solution

My solution presents an intelligent **PCA-based anomaly detection system** specifically designed for industrial process monitoring and fault detection. The system implements a sophisticated hybrid approach that combines Principal Component Analysis (PCA) with statistical **Z-score analysis** to provide comprehensive anomaly detection capabilities.

Key Components:

1. **Data Preprocessing Module**: Handles missing values, performs data validation, and ensures temporal consistency
2. **PCA-based Feature Extraction**: Reduces dimensionality while preserving critical variance patterns
3. **Hybrid Scoring Mechanism**: Combines PCA reconstruction error with statistical deviation scores
4. **Feature Attribution Engine**: Identifies the top contributing features for each anomaly
5. **Adaptive Scoring System**: Maps raw anomaly scores to interpretable percentile-based scores (0-100)

How it Addresses the Problem

The solution addresses critical challenges in industrial anomaly detection:

1. **Multi-dimensional Data Analysis**: Handles TEP_Train_Test.xlsx dataset with multiple correlated variables
2. **Real-time Monitoring**: Provides continuous scoring for ongoing process monitoring
3. **Interpretability**: Identifies specific features contributing to anomalies (top_feature_1 through top_feature_7)
4. **Adaptive Thresholding**: Uses training data distribution to establish dynamic anomaly thresholds

Innovation and Uniqueness of the Solution

Novel Hybrid Approach:

1. Combined PCA reconstruction error with statistical deviation metrics
2. Weighted combination (configurable via weight_pca and weight_z parameters)
3. Non-linear score compression to enhance sensitivity in critical ranges

Robust Data Handling:

1. Intelligent missing value imputation using forward-fill and interpolation
2. Automatic data type conversion and validation
3. Temporal index management for time-series data

2. Technical Approach

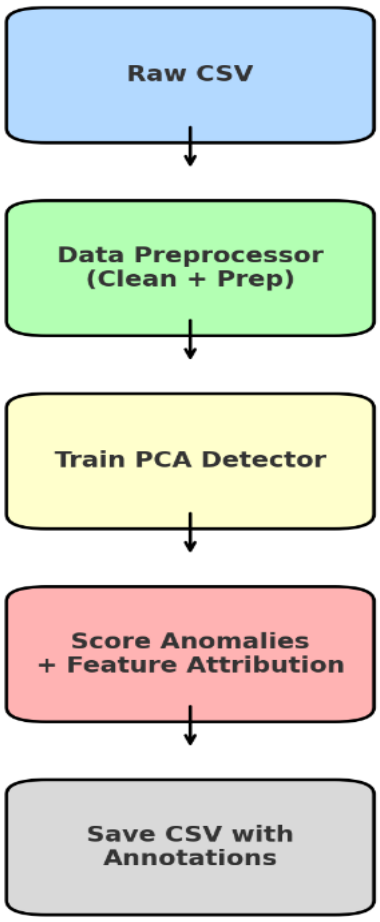
Technologies to be Used

Programming Languages & Core Libraries:

1. **Python 3.8+**: Primary development language
2. **NumPy**: Numerical computations and array operations
3. **Pandas**: Data manipulation and time-series handling
4. **Scikit-learn**: PCA implementation and machine learning utilities

Methodology

System Architecture Flow:



Implementation Process:

1. **Data Ingestion**: Load time-series industrial data with automatic timestamp detection.
2. **Preprocessing Pipeline**:
   a. Converted non-numeric data to numeric format
   b. Handled missing values using forward-fill and interpolation
   c. Established training and analysis time windows
3. **Model Training**:
   a. Calculated statistical parameters (mean, standard deviation)
   b. Trained PCA model with adaptive component selection (95% variance threshold)
   c. Established baseline anomaly score distribution
4. **Anomaly Detection**:
   a. Transform test data using trained parameters
   b. Calculate hybrid anomaly scores
   c. Perform feature attribution analysis
   d. Map scores to interpretable percentile scale
5. **Results Integration**: Merge anomaly scores and feature attributions with original dataset

Key Algorithms:

PCA Reconstruction Error:

$$\text{residual\_error} = ||X - \text{PCA\_inverse}(\text{PCA\_transform}(X))||^2$$

Hybrid Score Calculation:

$$\text{hybrid\_score} = \text{weight\_z} \times \sqrt{(\text{z\_score}^2)} + \text{weight\_pca} \times \sqrt{(\text{reconstruction\_error}^2)}$$

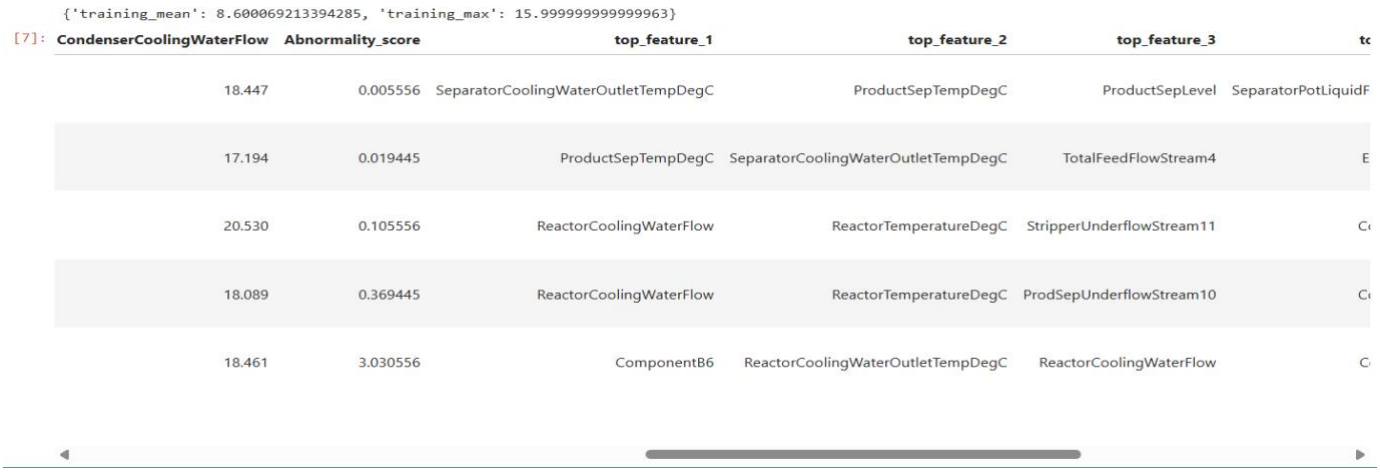
$$\text{z\_score}=(\text{x-mean})/(\text{standard deviation})$$

Feature Attribution:

$$\text{contribution\_i} = (\text{weight\_z} \times \text{z\_share\_i} + \text{weight\_pca} \times \text{residual\_share\_i})$$

Output

A. Training mean=8.6
B. Training max=15.99
C. Csv File with anomaly score and top 8 features column



{'training_mean': 8.600069213394285, 'training_max': 15.999999999999963}

[7]:

CondenserCoolingWaterFlow	Abnormality_score	top_feature_1	top_feature_2	top_feature_3	tc
18.447	0.005556	SeparatorCoolingWaterOutletTempDegC	ProductSepTempDegC	ProductSepLevel	SeparatorPotLiquidF
17.194	0.019445	ProductSepTempDegC	SeparatorCoolingWaterOutletTempDegC	TotalFeedFlowStream4	E
20.530	0.105556	ReactorCoolingWaterFlow	ReactorTemperatureDegC	StripperUnderflowStream11	C
18.089	0.369445	ReactorCoolingWaterFlow	ReactorTemperatureDegC	ProdSepUnderflowStream10	C
18.461	3.030556	ComponentB6	ReactorCoolingWaterOutletTempDegC	ReactorCoolingWaterFlow	C

3. Feasibility and Viability

Analysis of Feasibility

Technical Feasibility: HIGH

1. Built on proven mathematical foundations (PCA, statistical analysis)
2. Uses well-established Python libraries with strong community support
3. Modular design allows for incremental implementation and testing

Operational Feasibility: HIGH

1. Minimal computational requirements suitable for real-time deployment
2. Configurable parameters adapt to different industrial contexts
3. Clear output format (anomaly scores + feature attribution) for operator interpretation
4. Automated data preprocessing reduces manual intervention

Economic Feasibility: HIGH

1. Open-source technology stack minimizes licensing costs
2. Preventive maintenance capabilities can significantly reduce downtime costs
3. Scalable architecture supports multiple process monitoring applications
4. Low infrastructure requirements for deployment

Potential Challenges and Risks

Data Quality Risks:

A. **Challenge**: Inconsistent data quality, missing sensors, or corrupted measurements
B. **Impact**: Reduced detection accuracy and false positives

Model Drift:

A. **Challenge**: Industrial processes may change over time, affecting model performance
B. **Impact**: Degraded anomaly detection accuracy

Computational Scalability:

A. **Challenge**: Very high-dimensional datasets or real-time processing requirements
B. **Impact**: Potential latency in anomaly detection

Interpretability Complexity:

A. **Challenge**: Complex multi-feature interactions may be difficult to explain
B. **Impact**: Reduced operator trust and adoption

Strategies for Overcoming Challenges

Data Quality Management:

1. Implement robust data validation pipelines
2. Develop adaptive imputation strategies for different sensor types
3. Create data quality metrics and monitoring dashboards

Model Maintenance Strategy:

1. Implement automated model retraining schedules
2. Develop drift detection mechanisms
3. Create A/B testing frameworks for model updates
4. Establish performance monitoring and alerting systems

Performance Optimization:

1. Implement incremental PCA for large datasets
2. Utilize parallel processing for multi-process monitoring
3. Develop streaming data processing capabilities

Conclusion

This project successfully demonstrates a PCA-based anomaly detection framework for multivariate time series data. By training on a TEP_Train_Test dataset and applying the model across the full analysis window, the system assigns anomaly scores on a 0–100 scale and identifies the top contributing features for each detected anomaly.

The approach ensures:

1. Robust detection of both individual threshold violations and changes in variable relationships.
2. Interpretability through feature attribution, helping stakeholders understand the root causes of anomalies.
3. Scalability for large datasets, with preprocessing steps for missing values and timestamp alignment.

Overall, this solution enables a transition from reactive to proactive monitoring, supporting predictive maintenance and improving asset reliability. While PCA provides an efficient and interpretable baseline, the framework can be extended with advanced methods such as Isolation Forests, Autoencoders, or LSTM models for more complex temporal dependencies.

Future Work

While the current PCA-based framework provides a strong foundation for anomaly detection in multivariate time series, several enhancements can be pursued to improve accuracy, scalability, and practical applicability:

1. **Advanced Deep Learning Models**-Integrate LSTM Autoencoders or Variational Autoencoders (VAE) to capture complex temporal dependencies and non-linear feature interactions.

2. **Ensemble Methods**-Combine multiple algorithms (e.g., PCA, Isolation Forest, Autoencoder) into an ensemble system to reduce false positives and improve robustness.

3. **Real-Time Anomaly Detection**-Extend the solution to work in streaming environments using frameworks like Kafka or Spark, enabling real-time monitoring and immediate response.

4. **Adaptive Thresholding**-Implement dynamic thresholding that adapts anomaly score cutoffs based on seasonal or operating conditions, reducing unnecessary alerts.

Steps to Run the Code

1. Install Python 3.10 or above and ensure Jupyter Notebook is available.
2. Install the required libraries by running: **pip install numpy pandas scikit-learn**
3. Open the provided Jupyter Notebook file AnomalyDetection.ipynb in Jupyter Notebook or JupyterLab.
4. Place the input dataset file TEP_Train_Test.csv in the same directory as the notebook.
5. Run all the cells in the notebook sequentially from top to bottom.
6. After execution, a new file TEP_with_anomaly_output.csv will be generated in the same directory.
7. The output file will contain all original columns plus eight additional columns:
    A. Abnormality_score (0–100 scale)
    B. top_feature_1 … top_feature_7 (most contributing features or empty if none)
8. Open TEP_with_anomaly_output.csv to verify the results. Scores in the training window should remain low (mean < 10, max < 25), while anomalies in the analysis period will appear with higher scores.

References

[1] Ahmad et al. (2021). *Anomaly detection in time series: a comprehensive evaluation.* Journal of Big Data, SpringerOpen.
Available at: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00447-5

[2] BuiltIn (2025). *Step-by-Step Explanation of Principal Component Analysis (PCA).*
Available at: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[3] DataCamp (2025). *Principal Component Analysis in Python.*
Available at: https://www.datacamp.com/tutorial/principal-component-analysis-in-python

[4] Analytics Vidhya (2024). *Anomaly Detection Using PCA – Unveiling Insights in Data Anomalies*.
Available at: https://www.analyticsvidhya.com/blog/2024/04/anomaly-detection-using-pca-unveiling-insights-in-data-anomalies/

[5] Visual Studio Magazine (2021). *Anomaly Detection with PCA.*
Available at: https://visualstudiomagazine.com/articles/2021/10/20/anomaly-detection-pca.aspx

[6] Kaggle (2025). *Anomaly Detection Techniques Summary (PCA Section).*
Available at: https://www.kaggle.com/code/praxitelisk/anomaly-detection-techniques-summary#PCA