Multivariate Time Series Anomaly Detection

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.

Section	Explanation
Goal	Build a Python program to detect anomalies (abnormal behavior) in time series data that has multiple columns (i.e., multivariate).
Input	A CSV file with: 1) Multiple columns of numerical time series data 2) A defined time range or label for normal data (used for training) Modify the same CSV by adding exactly 8 new columns: 1) Abnormality score: Float values from 0.0 to 100.0 2) top_feature_1 through top_feature_7: String values containing original column names 3) If fewer than 7 features contribute, fill remaining columns with empty strings
Output	For each row, identify which 7 columns (features) contributed the most to the abnormal behavior: 1) Use model-specific methods 2) Rank by absolute contribution magnitude 3) Break ties using alphabetical order 4) Only include features contributing >1% to the anomaly 5) Fill remaining slots with empty strings if fewer than 7 contributors
Definition: Time Series Data	Data that is collected at regular intervals over time. Example: temperature readings every hour, pressure measurements every minute.
Definition: Multivariate	Data with multiple variables being recorded at each time point. 1) Example: temperature, humidity, pressure, vibration, flow rate all measured simultaneously.
Definition: Normal Period	A period in the data where everything was working fine (no abnormalities). This is used to train the model. 1) For this Hackathon: Normal Period for Model Training is 1/1/2004 0:00 to 1/5/2004 23:59 (120 hours of data).

What is Anomaly Detection?	**Identifying values in the data that don't behave like the normal period. These are "weird" or "unexpected" patterns. 1) For this Hackathon: Period for Model Execution and searching for Anomalies is 1/1/2004 0:00 to 1/19/2004 7:59 (439 hours of data). 2) Important: Overlap with the Training period is deliberate. Anomaly scores should be close to 0 (< 10) for the training period to validate model correctness.**

Type of Anomalies to Detect	**1) Threshold Violation – Individual variables exceeding their normal statistical ranges (e.g., > 3 standard deviations from training mean) 2) Relationship Change – Variables no longer following their usual correlations (e.g., temperature and humidity normally correlate at r=0.8, but suddenly show r=0.2) 3) Pattern Deviations – Temporal sequences that differ from normal operational patterns**

Degree of Abnormality (0-100 Scale)	**1) 0-10: Normal behavior (expected for training period) 2) 11-30: Slightly unusual but acceptable 3) 31-60: Moderate anomaly requiring attention 4) 61-90: Significant anomaly needing investigation 5) 91-100: Severe anomaly requiring immediate action 6) Calculation Method: Transform model outputs using percentile ranking within the analysis period.**

Top Contributors Calculation	**For each row, identify which 7 columns (features) contributed the most to the abnormal behavior: 1) Use model-specific methods 2) Rank by absolute contribution magnitude 3) Break ties using alphabetical order 4) Only include features contributing >1% to the anomaly 5) Fill remaining slots with empty strings if fewer than 7 contributors**

Data Quality Handling (Optional in this Hackathon since the data provided is of good quality)	Optional - your code should handle: 1) Missing values: Use forward-fill or linear interpolation 2) Invalid data: Replace non-numerical values with last good values 3) Timestamp validation: Ensure regular intervals 4) Constant features: Handle features with zero variance 5) Error reporting: Provide clear error messages for data issues

---

Steps to Build	1) Data Preprocessing: Load CSV, validate timestamps, handle missing values 2) Data Splitting: Separate normal period (training) from full analysis period 3) Model Training: Train anomaly detection model on normal data only 4) Anomaly Detection: Apply model to entire analysis period 5) Score Calculation: Transform model outputs to 0-100 scale 6) Feature Attribution: Calculate contributing features for each row 7) Output Generation: Add 8 new columns to original CSV file

ML Technique Suggested	Primary Options: 1) Isolation Forest: Good for global anomalies, built-in feature importance 2) Autoencoders: Learn complex patterns, use reconstruction error 3) PCA-based: Dimensionality reduction with reconstruction error Advanced Options: 1) LSTM Autoencoders: For temporal pattern learning 2) Ensemble Methods: Combine multiple techniques Potential Libraries: 1) Sklearn 2) Tensorflow 3) Keras 4) Pytorch 5) or any Python library

Code Structure Requirements	**1) Main function: Accept input_csv_path and output_csv_path as parameters 2) Modular design: Separate classes for data processing, model training, prediction 3) Type hints: Include type annotations for all functions 4) Documentation: Docstrings for all classes and functions 5) PEP8 compliance: Follow Python style guidelines 6) Error handling: Graceful handling of common issues**

Expected Columns in Output CSV	**1) All original columns 2) Abnormality_score (0 to 100) 3) top_feature_1, top_feature_2, ..., top_feature_7 (names of top contributing columns)**

Success Criteria	**Functional Requirements: 1) Code runs without errors on test dataset. 2) Produces all required output columns 3) Code runs without errors on other datasets that are formatted identical to given test dataset. 4) Training period anomaly scores: mean < 10, max < 25 Technical Quality: 1) Follows PEP8 standards 2) Modular, documented code 3) Handles edge cases appropriately Performance Validation: 1) Feature attributions make logical sense 2) No sudden score jumps between adjacent time points 3) Reasonable runtime (< 15 minutes for typical datasets)**

Edge Cases to Handle	**1) All normal data: Should produce low scores (0-20 range) 2) Training period anomalies: Warn user but proceed with training 3) Insufficient data: Require minimum 72 hours of training data 4) Single feature dataset: Handle cases with <7 features 5) Perfect predictions: Add small noise to avoid exactly 0 scores 6) Memory constraints: Handle datasets up to 10,000 rows efficiently**

Deliverables	**1) Python script(s): Complete, executable solution 2) Modified CSV file: Original data with 8 new columns 3) Sample usage: Example of how to run the code**

Evaluation	**1) The code will be executed by the person evaluating. 2) The code must run and generate the output specified above. 3) The code should be modular and follow PEP8.**

Download the file from here: TEP_Train_Test.csv

Your output needs to be uploaded as pdf file(s) only, using the "Upload Files" option below.

You also need to provide an outline of your submission in the text box below. The outline should contain:

1. Proposed Solution (Describe your Idea/Solution/Prototype):

- Detailed explanation of the proposed solution
- How it addresses the problem
- Innovation and uniqueness of the solution

2. Technical Approach:

- Technologies to be used (e.g. programming languages, frameworks, hardware)
- Methodology and process for implementation (Flow Charts/Images/ working prototype)

3. Feasibility and Viability:

- Analysis of the feasibility of the idea
- Potential challenges and risks
- Strategies for overcoming these challenges

4. Research and References:

- Details / Links of the reference and research work

Write your answer here