Practical Lab 1 - Streaming Data for Predictive Maintenance with Linear Regression-Based Alerts

Hide Assignment Information

Instructions

📂 Practical Lab 1: Streaming Data for Predictive Maintenance with Linear Regression-Based Alerts

Context

In the **Data Stream Visualization Workshop**, you learned how to stream, store, and visualize industrial current data.
This lab extends that work: you will now implement **regression-based anomaly detection** to generate **alerts and errors** when currents deviate significantly from their expected values.

This task simulates a **Predictive Maintenance** scenario, where early alerts and errors can flag potential failures before they occur.

Learning Objectives

By the end of this assignment, you will be able to:

- Extend a streaming pipeline with **machine learning models**.
- Apply **linear regression** to detect unusual consumption trends.
- Analyze regression residuals and discover meaningful thresholds for anomaly detection.
- Implement an **alerts/errors module** in a streaming context.

Submit Cancel

Deliverables

Your GitHub repository must contain:

1. **README.md**
    - Professional and well-structured.
    - Project summary.
    - Clear setup instructions.
    - Explanation of regression/alert rules.
    - Screenshots or plots of results.
2. **requirements.txt**
    - All dependencies with versions.
3. **Data folder**
    - Relevant CSV files used for training (e.g., `RMBR4-2_export_test.csv`).
4. **Codebase**
   Must include scripts and/or Jupyter Notebooks that:
    - Connect to a **cloud-based database** (e.g., Neon.tech PostgreSQL). Use the database to pull the information in it to the application and train the Linear Regression model.
    - Ingest and query streaming data. You can read the testing data from the synthetically generated data in a Dataframe or from a CSV file.
    - Run regression models (Time → Axes #1–#8).
    - Implement alerts/errors based on thresholds you discover.

🔍 Discovering Thresholds

You are not given fixed thresholds. Instead, you must **discover them from the data**.

To do this:

1. **Develop regression models**:

- For each axis (#1–#8), fit (**train**) a univariate linear regression (Time → Axis values) and produce a model able to make predictions when fed **testing** data (read below)
- Record slope and intercept.
- Plot scatter data with regression lines.

2. **Analyze residuals**:
    - Compute the difference between observed values and the regression prediction.
    - Plot distributions (scatter, line, or boxplots of residuals).
    - Look for outliers and patterns.
3. **Define thresholds**:
    - Choose **MinC**: the minimum current deviation (kWh above regression line) that should trigger an **Alert** if sustained.
    - Choose **MaxC**: the maximum current deviation (kWh above the regression line) that should trigger an **Error** if sustained.
    - Choose **T**: the minimum continuous time (in seconds) that the deviation must persist.
4. **Implement alert/error rules**:
    - **Alert:** ≥ **MinC** kWh above regression line for ≥ **T** seconds continuously
    - **Error:** ≥ **MaxC** kWh above regression line for ≥ **T** seconds continuously
5. **Produce testing data *synthetically***
    - Use the **training metadata** to synthetically generate **testing** data to run predictions using the linear regression model.
6. **Visualize alerts/errors**:
    - Overlay Alert/Error markers on your regression plots.
    - Annotate each with its duration.
7. **Log results**:
    - Store Alert and Error events in a structured CSV or database table.

👉 **Hint:** To produce the testing data:

- Upload the training data to an LLM
- Write a prompt to request that the LLM generate a new file with the same metadata, and a similar mean/std. deviation.
- Normalize the data set, if necessary.
    - Min-Max Normalization Algorithm
- Standardize the data set, if necessary
    - Use Z-scores to find proportions

👉 **Hint:** Follow the process we explored in the workshop extension:

- Fit regression models across all axes,
- Compare residuals and outlier counts,
- Test different thresholds and time windows until alerts/errors appear,
- Interpret your results in the context of **Predictive Maintenance**.

You will be graded not only on whether your code works, but also on **how well you justify your chosen MinC, MaxC, and T values** using evidence from your analysis.

Submission Instructions

- Push your completed project to your GitHub repository.
- Submit a PDF with your name, student IDs, and the URL to your repository via the course website.
- Your GitHub repository must include all deliverables listed above.

📝 Marking Rubric (10 points total)

Category	Criteria	Points
Project Setup	Repo includes `README.md`, `requirements.txt`, and `data/`. README files that are professional and clear. The IPYNB file has the output of the last test run before submitting the project.	1
Database Integration	Code connects to Neon.tech (or equivalent PostgreSQL) and ingests/queries streaming data.	1.5
Streaming Simulation	Script simulates CSV → DB time-based flow using synthetic test data that has been properly	1

Category	Criteria	Points
	normalized and standardized with respect to the training data from the relational database.	
Regression Models & Residual Analysis	Fits univariate regressions (Time → Axes #1–#8), plots regression lines, and analyzes residuals.	2
Threshold Discovery & Justification	Student justifies chosen **MinC, MaxC, T** with evidence from analysis (plots, residuals, predictive maintenance context).	2
Alerts & Errors Implementation	Correctly implements detection logic with chosen thresholds. Logs events.	2
Visualization/Dashboard	Shows regression with alert/error annotations. Clear plots.	0.5

Total: 10 points

✅ This assignment is a **follow-up** to the Data Streaming Workshop:

- You reuse the pipeline they built,
- Extend it with regression + thresholds,
- And must **think critically** about thresholds instead of copying fixed values.

Submissions past the due date:

- 10% deduction after 11:00 AM pm on the due date.
- 25% deduction after 11:59 pm on the due date.
- No credit after 11:59 pm on the day after the due date.

Academic integrity:

- **The instructor may summon students suspected of academic dishonesty for a code review to verify their understanding and ability to reproduce any stage of the development process. The instructor reserves the right to deduct marks, either partially or entirely.**

Due on Feb 6, 2026 11:59 PM

Available on Jan 9, 2026 12:01 AM. **Access restricted before availability starts.**

Available until Feb 8, 2026 11:59 PM. **Access restricted after availability ends.**

Submit Assignment

Allowed File Extensions

pdf

Files to submit

(0) file(s) to submit

After uploading, you must click Submit to complete the submission.

Add a File **Record Audio** **Record Video**

Comments