ML-Based Anomaly Detection with Concept Drift

IEEE-Style Project Documentation and Case Study

Muhammad Haider Abbas | MSc Computer Science & Technology | Ulster University Birmingham Campus



Figure 1. Project visual identity used in the portfolio website.

Abstract

ML-Based Anomaly Detection with Concept Drift is documented as a professional portfolio case study by Muhammad Haider Abbas. The project demonstrates practical capability in AI / Machine Learning through a structured problem-solution design, implementation direction, testing plan and future development roadmap. The report explains the business or academic motivation, system requirements, architecture, workflow, quality considerations and expected professional value of the project. It is prepared for use as downloadable portfolio evidence for recruiters, clients and academic reviewers.

Keywords: AI / Machine Learning, Academic Research, Researcher & Developer, Python, Scikit-learn, River, Streaming ML

I. Project Profile

Field	Details
Project Title	ML-Based Anomaly Detection with Concept Drift
Category	AI / Machine Learning
Year	2026
Role	Researcher & Developer
Status	Academic Research
Tools	Python, Scikit-learn, River, Pandas, Matplotlib
Portfolio Slug	ml-anomaly-detection-concept-drift

II. Introduction and Problem Context

This document presents the technical and professional documentation for ML-Based Anomaly Detection with Concept Drift. The project is positioned as a academic research in the AI / Machine Learning category. The work highlights the author's ability to combine technical implementation, documentation discipline and outcome-focused project thinking. The project summary is: A streaming machine learning research project focused on detecting anomalies when behaviour changes over time.

A. Problem Statement

Static anomaly detectors can become unreliable when the underlying data distribution changes. In security, finance, infrastructure and IoT systems, this can lead to missed threats or excessive false alarms.

B. Proposed Solution

The project uses streaming data preparation, model training, drift monitoring and comparative evaluation to identify when models need adaptation.

III. Aim and Objectives

To design a drift-aware anomaly detection pipeline that can remain useful when normal behaviour changes over time in streaming data.

- Review anomaly detection and concept drift issues in non-stationary data.
- Prepare a streaming-style dataset and define baseline outlier detection models.
- Compare batch and adaptive approaches using precision, recall, F1-score, false alarm rate and detection delay.
- Design an evaluation workflow suitable for MSc-level research and future implementation.

IV. Requirements Analysis

ID	Requirement	Type	Priority
FR-01	Import and clean chronological data	Functional / project requirement	Required
FR-02	Create baseline model using scikit-learn	Functional / project requirement	Required
FR-03	Simulate streaming updates using River	Functional / project requirement	Required
FR-04	Detect or flag behaviour shift	Functional / project requirement	Required
FR-05	Visualise anomaly and drift points	Functional / project requirement	Required
NFR-01	Usability and clear portfolio presentation	Non-functional	Required
NFR-02	Maintainability and future extensibility	Non-functional	Required
NFR-03	Professional documentation and evidence quality	Non-functional	Required

V. System Architecture

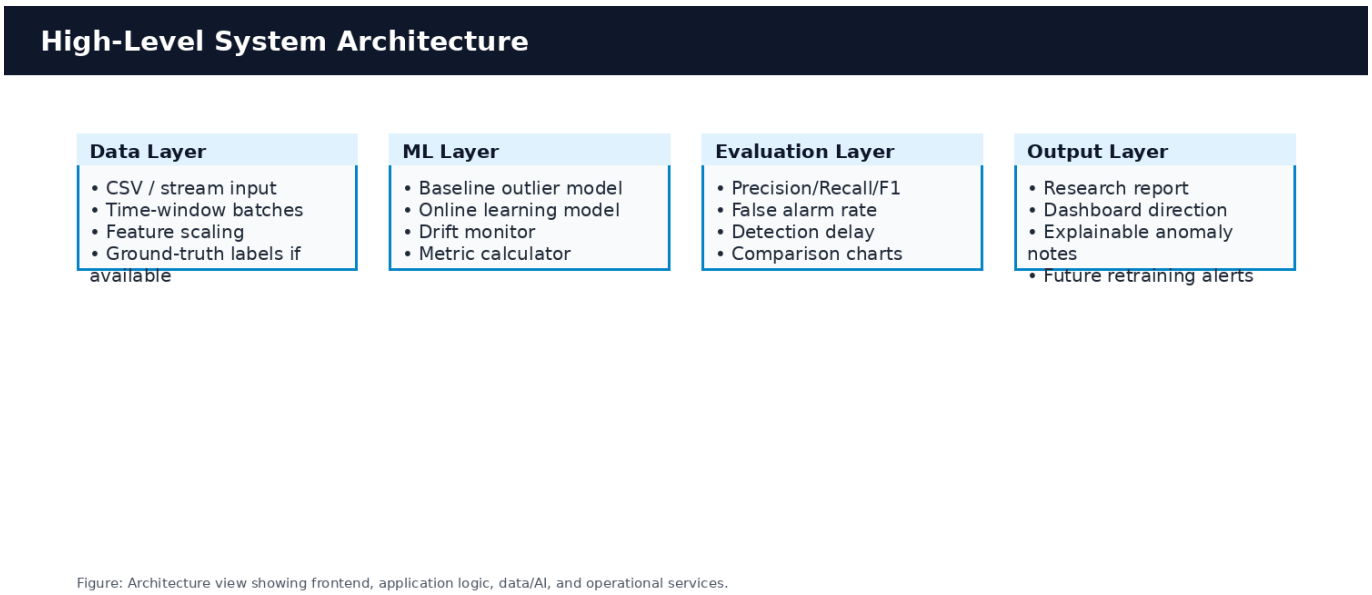


Figure: Architecture view showing frontend, application logic, data/AI, and operational services.

Figure 2. High-level architecture or production pipeline for the project.

The architecture separates user-facing presentation, project logic, data or model processing, and operational output. This structure makes the project easier to explain, test and extend inside a professional portfolio environment.

VI. Workflow Design

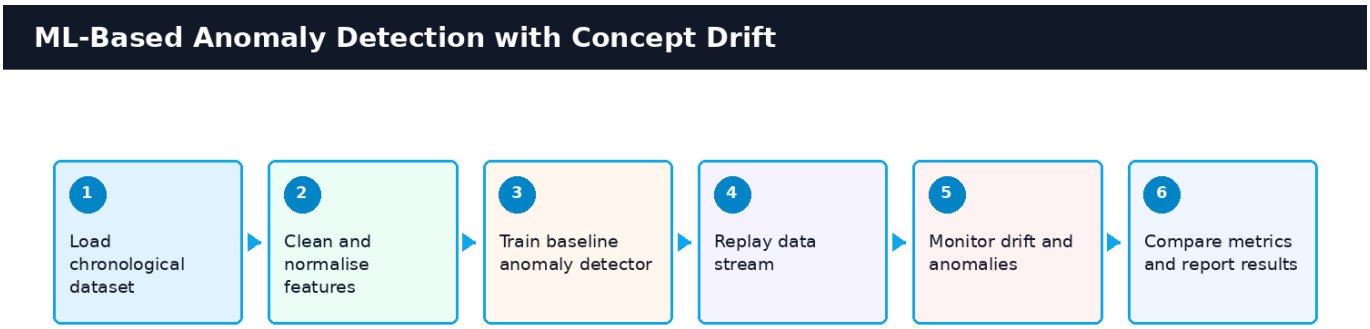


Figure: Generated project workflow / architecture artefact for portfolio documentation.

Figure 3. Project workflow from input or user action to final output.

VII. Data, Components and Implementation View

Component	Layer	Responsibilities	Notes
C-01	Data Layer	CSV / stream input, Time-window batches, Feature scaling, Ground-truth labels if available	Core component group
C-02	ML Layer	Baseline outlier model, Online learning model, Drift monitor, Metric calculator	Core component group
C-03	Evaluation Layer	Precision/Recall/F1, False alarm rate, Detection delay, Comparison charts	Core component group
C-04	Output Layer	Research report, Dashboard direction, Explainable anomaly notes, Future retraining alerts	Core component group

Implementation should be documented through screenshots, code snippets, database schema extracts, test outputs or demonstration media when available. For deployed or live systems, screenshots and logs should be attached as appendices to increase evidence strength.

VIII. Testing and Validation Strategy

Test Area	Test Method	Expected Result
Data validation	Missing values, duplicates and feature ranges checked	Pass when data is consistent
Baseline model	Run static outlier detector	Produces reproducible anomaly labels
Streaming simulation	Process records in chronological order	No future leakage
Metric reporting	Calculate precision, recall and F1	Metrics appear in final table

The testing strategy focuses on validating the main user or system journey, confirming that core requirements behave as expected, and recording limitations clearly. This supports honest portfolio evidence and avoids overclaiming features that are still prototypes.

IX. Security, Ethics and Professional Considerations

- Avoid presenting AI recommendations as guaranteed truth.
- Protect user data and avoid uploading sensitive CV, health or behavioural data without consent.
- Use evaluation metrics and explain limitations clearly.

X. Results, Outcome and Future Work

A. Outcome

A strong MSc project foundation suitable for research, implementation, evaluation and dissertation writing.

B. Future Enhancements

Add dashboard visualisation, live stream simulation, adaptive model retraining and explainable anomaly reports.

Roadmap Stage	Enhancement Direction
Short term	Add missing screenshots, demo links and downloadable supporting files.
Medium term	Connect live/source URLs and replace placeholder evidence with real artefacts.
Long term	Use analytics, user feedback or formal evaluation results to strengthen the project case study.

XI. References

[1] Montiel et al., "River: machine learning for streaming data in Python," https://arxiv.org/abs/2012.04740

[2] scikit-learn, "Novelty and Outlier Detection," https://scikit-learn.org/stable/modules/outlier_detection.html

Appendix A. Portfolio Download Mapping

Item	Value
Recommended PDF path	docs/ml-anomaly-detection-concept-drift-project-documentation.pdf
Recommended Word path	docs/ml-anomaly-detection-concept-drift-project-documentation.docx
Admin field: Documentation	docs/ml-anomaly-detection-concept-drift-project-documentation.pdf
Admin field: Extra Project File	docs/ml-anomaly-detection-concept-drift-project-documentation.docx