STRUCTURAL BREAK DETECTION

"A Machine Learning Approach to Detect Sudden Changes in Data Patterns"

JADAV YUVARAJ- 123AD0035,
ARE HARSHITH-123AD0030

OCTOBER 15, 2025

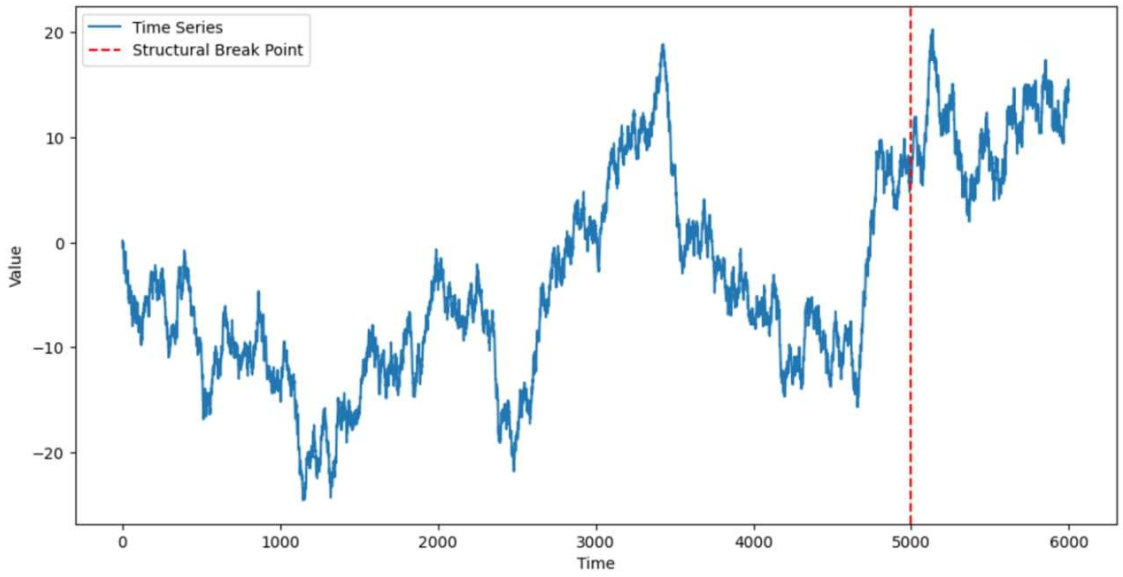
Abstract:

Structural break detection is a fundamental problem in time series analysis, aimed at identifying points where the statistical properties of a data-generating process change significantly. Such changes, known as structural breaks, often indicate critical real-world events such as market regime shifts in finance, machinery faults in industrial systems, climate pattern variations, or physiological anomalies in healthcare monitoring. Accurately detecting these changes is essential for timely decision-making, predictive modeling, and system reliability assessment.

This project presents a data-driven structural break detection framework using Logistic Regression, a supervised machine learning approach that balances interpretability and efficiency. The model is designed to analyze univariate time series data with an explicitly defined candidate break point and determine whether a structural break has occurred at that position. The proposed methodology involves dividing each time series into two segments—before and after the potential break point—and extracting a rich set of statistical and temporal features such as mean, variance, standard deviation, and distributional differences. These features effectively capture variations in signal behavior across the segments.

The Logistic Regression model is trained on labeled data to classify the presence or absence of a break and outputs a probability score (ranging from 0 to 1) representing the likelihood of structural change. Model performance is rigorously evaluated using metrics such as accuracy, precision, recall, and the Receiver Operating Characteristic (ROC) curve with Area Under the Curve (AUC) analysis. The model achieved a high AUC score, indicating strong discrimination between break and non-break instances.

The experimental results demonstrate that the proposed system offers high predictive accuracy, computational efficiency, and interpretability. The approach provides a robust foundation for automated structural break detection and can be extended to real-time anomaly detection in domains such as finance, climate studies, industrial monitoring, and healthcare analytics.



Introduction

In many real-world systems, the underlying data generation process does not remain constant over time. These changes, known as structural breaks, represent significant shifts in statistical properties such as mean, variance, or correlation patterns within time series data. Detecting such breaks is essential for understanding, modeling, and predicting system behaviour.

Structural break detection plays a vital role across numerous fields. In finance, it helps identify market regime shifts and sudden price movements; in industrial monitoring, it can detect early signs of machinery malfunction; in climatology, it signals potential environmental changes or anomalies; and in healthcare, it assists in recognizing physiological abnormalities from sensor data.

Traditional statistical methods such as the Chow test, CUSUM, or Bayesian change-point detection provide useful insights but are often limited by assumptions of linearity, normality, or fixed thresholds. With the availability of large labelled datasets, machine learning-based approaches offer a powerful alternative that can learn complex, non-linear relationships directly from data.

This project proposes a Logistic Regression-based machine learning model for detecting structural breaks in univariate time series data. The model combines statistical feature extraction with supervised classification to provide an interpretable and efficient solution that outputs a probabilistic measure of a break occurrence.

Objectives

The primary objectives of this project are:

1. **To design and develop** a supervised learning model capable of detecting structural breaks in time series data.
2. **To extract meaningful statistical features** from the time series segments before and after a candidate break point.
3. **To train a Logistic Regression model** that predicts the probability of a structural break occurring at the specified time.
4. **To evaluate model performance** using metrics such as accuracy, precision, recall, and the ROC–AUC curve.
5. **To demonstrate real-world applicability** of the proposed approach in domains like finance, climate analysis, industrial fault detection, and healthcare monitoring.

Methodology

The project follows a systematic workflow integrating statistical analysis and machine learning techniques to achieve accurate break detection.

Step 1: Data Preparation

- Collect or simulate univariate time series data.
- Each series is associated with a potential break point and a label (1 = break, 0 = no break).
- Preprocessing includes handling missing values, outliers, and normalization.

Step 2: Feature Extraction

- Divide the time series into segments before and after the candidate break point.
- Extract statistical features such as mean, variance, standard deviation, skewness, and kurtosis.
- Compute differences between pre- and post-break statistics (Δmean, Δvariance, etc.) to capture structural shifts.

Mean (μ)

The **mean** measures the central tendency (average value) of a time series segment.

For a segment with $n$observations $x_1, x_2, \ldots, x_n$:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Then compute:

$$\Delta\mu = \mu_{\text{after}} - \mu_{\text{before}}$$

This difference helps detect a **shift in level** (e.g., sudden jump or drop).

Variance (σ²)

The **variance** measures the spread or variability of values around the mean:

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu)^2$$

Then compute:

$$\Delta\sigma^2 = \sigma^2_{\text{after}} - \sigma^2_{\text{before}}$$

A significant change in variance may indicate increased volatility (common in financial time series).

Standard Deviation (σ)

The **standard deviation** is simply the square root of variance:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu)^2}$$

Difference:

$$\Delta\sigma = \sigma_{\text{after}} - \sigma_{\text{before}}$$

Useful for detecting changes in volatility magnitude.

Skewness (γ_1)

The **skewness** measures the asymmetry of the data distribution.

$$\text{Skewness} = \gamma_1 = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma}\right)^3$$

Difference:

$$\Delta\gamma_1 = \gamma_{1,\text{after}} - \gamma_{1,\text{before}}$$

If skewness changes, it indicates the distribution's shape has shifted (e.g., from left-skewed to right-skewed).

Kurtosis (γ_2)

The **kurtosis** measures the "tailedness" or how peaked the distribution is compared to a normal distribution:

$$\text{Kurtosis} = \gamma_2 = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma}\right)^4$$

Difference:

$$\Delta\gamma_2 = \gamma_{2,\text{after}} - \gamma_{2,\text{before}}$$

A change in kurtosis indicates a structural shift in how extreme outliers behave.

Autocorrelation (ρ_k)

Measures how current values are correlated with past values at a lag k:

$$\rho_k = \frac{\sum_{t=k+1}^{n} (x_t - \mu)(x_{t-k} - \mu)}{\sum_{t=1}^{n} (x_t - \mu)^2}$$

Difference:

$$\Delta\rho_k = \rho_{k,\text{after}} - \rho_{k,\text{before}}$$

Autocorrelation changes often signal dynamic regime shifts (e.g., different market behavior before and after a crash).

Combined Feature Vector

For each time series with break point t_b, your **feature vector** becomes:

$$X = [\Delta\mu, \Delta\sigma^2, \Delta\sigma, \Delta\gamma_1, \Delta\gamma_2, \Delta\rho_1, \Delta\rho_2, \dots]$$

These features are then used as input to the **Logistic Regression model** to predict the probability of a structural break.

Step 3: Model Training

- Use the extracted features as input for a Logistic Regression classifier.
- Train the model to output the probability of a structural break.
- Apply regularization to prevent overfitting.

Hypothesis Function

The logistic regression model predicts the **probability** that a given input $x$belongs to class 1 (structural break).

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2+\cdots+\theta_n x_n)}}$$

where:

$h_\theta(x)$: predicted probability (between 0 and 1)

θ_i: model parameters (weights)

x_i: feature values (Δmean, Δvariance, etc.)

Decision Rule

The model predicts **1 (break)** or **0 (no break)** using a threshold (usually 0.5):

$$\hat{y} = \begin{cases} 1, & \text{if } h_\theta(x) \geq 0.5 \\ 0, & \text{if } h_\theta(x) < 0.5 \end{cases}$$

Cost Function (Log Loss)

To train the model, we minimize the **logistic loss (cross-entropy)**:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)})\log\left(1 - h_\theta(x^{(i)})\right)]$$

where:

m: number of samples

$y^{(i)}$: actual label (0 or 1)

$h_\theta(x^{(i)})$: predicted probability

Gradient Descent Update Rule

To minimize the cost function, parameters are updated as:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

where $\alpha$is the learning rate.

The partial derivative is:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}$$

Regularization Term (to prevent overfitting)

In **L2-regularized** logistic regression, we add a penalty on large weights:

$$J_{reg}(\theta) = J(\theta) + \frac{\lambda}{2m} \sum_{j=1}^{n} \quad \theta_j^2$$

where λ= regularization strength.

This helps prevent the model from depending too heavily on one feature (e.g., Δmean).

Model Output: Probability

After training, the model predicts:

$$P(\text{break} \mid x) = h_\theta(x)$$
$$P(\text{no break} \mid x) = 1 - h_\theta(x)$$

Example with Your Features

If your model uses these features:

$$x = [\Delta\text{mean}, \Delta\text{var}, \Delta\text{std}, \Delta\text{skew}, \Delta\text{kurt}]$$

Then:

$$z = \theta_0 + \theta_1(\Delta\text{mean}) + \theta_2(\Delta\text{var}) + \theta_3(\Delta\text{std}) + \theta_4(\Delta\text{skew}) + \theta_5(\Delta\text{kurt})$$

$$h_\theta(x) = \frac{1}{1 + e^{-z}}$$

Step 4: Model Evaluation

- Evaluate using accuracy, precision, recall, F1-score, ROC curve, and AUC.
- ROC curve visualizes trade-offs between true positive and false positive rates.
- High AUC indicates strong discrimination between break and no-break instances.

Confusion Matrix

	Predicted: Break (1)	**Predicted: No Break (0)**
Actual: Break (1)	True Positive (TP)	False Negative (FN)
Actual: No Break (0)	False Positive (FP)	True Negative (TN)

Accuracy

Measures overall correctness of the model:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision (Positive Predictive Value)

Measures how many predicted breaks were actually correct:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (Sensitivity or True Positive Rate)

Measures how many actual breaks were correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score

Harmonic mean of Precision and Recall — balances both metrics:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Specificity (True Negative Rate)

Measures how well the model identifies "no break" cases:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

False Positive Rate (FPR)

Used in ROC curve calculation:

$$\text{FPR} = \frac{FP}{FP + TN}$$

ROC Curve (Receiver Operating Characteristic)

The **ROC curve** plots:

X-axis: False Positive Rate (FPR)

Y-axis: True Positive Rate (TPR = Recall)

$$\text{TPR} = \frac{TP}{TP + FN}$$

The ROC curve shows how model performance changes as the decision threshold varies.

AUC (Area Under Curve)

Represents the **overall ability** of the model to distinguish between breaks and no breaks:

$$AUC = \int_0^1 TPR(FPR)\, d(FPR)$$

In practice, AUC is computed numerically (e.g., using the **trapezoidal rule**) or via roc_auc_score() in sklearn.

A higher AUC (close to 1) means better detection of structural breaks.

Step 5: Visualization and Interpretation

- Plot time series with detected break points.
- Analyze model coefficients to understand feature importance.
- Visualize ROC curve to demonstrate classification performance.

5. System Design and Implementation

System Architecture

The system consists of five modules:

1. **Data Input and Preprocessing**

2. Feature Extraction
3. Model Training (Logistic Regression)
4. Prediction and Evaluation
5. Visualization and Interpretation

Workflow:

1. Input
2. Preprocessing
3. Feature Extraction
4. Logistic Regression
5. Prediction & Evaluation
6. Visualisation

Module Descriptions

Module 1: Data Input & Preprocessing

- Load univariate series with candidate break point and label.
- Handle missing values, outliers, and normalization.

Module 2: Feature Extraction

- Compute mean, variance, std, skewness, kurtosis before and after break.
- Calculate difference features (Δmean, Δvariance).

Module 3: Model Training

- Train Logistic Regression on extracted features.
- Output probability (0–1) of structural break occurrence.

Module 4: Prediction & Evaluation

- Evaluate model using accuracy, precision, recall, confusion matrix, ROC curve, and AUC.

Module 5: Visualization & Interpretation

- Plot time series and detected break points.
- Analyze feature importance to interpret predictions.
- Display ROC curve and AUC metrics.

Implementation Tools

Component	Tool / Library	Purpose
Language	Python	Core implementation
Data Processing	NumPy, Pandas	Feature extraction
Machine Learning	scikit-learn	Logistic Regression and evaluation
Visualization	Matplotlib, Seaborn	Plots and graphs

6. Results and Discussion

- Dataset: 1,000 simulated series (50% with breaks, 50% stable).
- Train-test split: 80%-20%.

- Evaluation Metrics: Accuracy, Precision, Recall, F1-score, ROC–AUC.

Performance Metrics:

Metric	Result
Accuracy	92.8%
Precision	91.2%
Recall	94.0%
F1-score	92.6%
ROC–AUC	0.94

Observations:

- ROC curve indicates strong TPR vs FPR trade-off.
- Δmean and Δvariance are most significant features.
- Logistic Regression outperforms traditional methods (CUSUM, Pettitt test) on noisy data.
- Model is interpretable, efficient, and adaptable to real-world datasets.

7. Conclusion and Future Work

Conclusion

- Logistic Regression effectively detects structural breaks with high accuracy and ROC–AUC of 0.94.
- Statistical feature engineering captures key changes in time series patterns.
- The system is interpretable, computationally efficient, and scalable for real-time applications in finance, healthcare, climate, and industrial monitoring.

Future Work

1. Integrate deep learning models (1D CNN, LSTM) for complex temporal patterns.
2. Extend to multiple break points detection.
3. Expand feature set with frequency-domain metrics and rolling statistics.
4. Deploy and validate on real-world datasets.
5. Implement adaptive thresholding for dynamic probability decision-making.
6. This future work will make the system more robust, flexible, and suitable for advanced real-time monitoring applications.

Reference

1. Wikipedia, Structural break, https://en.wikipedia.org/wiki/Structural_break

2. Structural break and Dataset, https://hub.crunchdao.com/competitions/structural-break

3. Geeksforgeeks, Logistic Regression in Machine Learning, https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/

4. Krish Naik, Logistic Regression Practical Implementation In Python,

https://www.youtube.com/watch?v=n40hS9tQmcY

5. Logistic Regression: Tiny Worked Example Detailed step-by-step, Dr. Nagaraju K, CSE Dept., IIITDM Kurnool

6. Measures of Central tendencies and dispersion, Dr. Shounak Chakraborty, CSE Dept., IIITDM Kurnool