**Anomaly Detection Pipeline - Detailed Documentation**

# Executive Summary

This document provides a comprehensive description of the anomaly detection pipeline implemented in this repository. It covers the data schema, modeling approach (Negative Binomial NB2), model selection, change-point handling, diagnostics, and visualization. An embedded example plot and a slice of diagnostics are included for quick validation.

# Data Schema and Preparation

Expected input CSV columns:

- insight\_date\_time (datetime)

- target\_type (string)

- insight\_type (string)

- target\_key (string)

- other columns (optional)

Weekly aggregation: each record is converted to the start-of-week bucket using Monday as week start. Missing weeks are filled with zero counts for each (target\_type, insight\_type, target\_key).

# Modeling Details

We model weekly counts using a Negative Binomial NB2 (mean-variance relationship Var(y)=mu+alpha\*mu^2). Key points:

- Linear predictor: intercept, optional Fourier terms for seasonality (K harmonics), and a light linear trend (t / 52).

- Multiple candidate models are fit per series varying fourier\_K; best model is chosen by AIC.

Prediction intervals are computed by transforming the model-predicted mean into NB parameters and using the Negative Binomial quantiles (scipy.stats.nbinom.ppf). This yields integer-aware PI appropriate for counts.

# Change Point Detection

We detect structural changes using the ruptures library with the PELT algorithm and an l2 cost on log1p-transformed counts. Steps:

- Transform counts with log1p for stability.

- Run rpt.Pelt(model="l2").fit(transformed\_series) and predict with a selected penalty.

- If change points are found, split the series and fit models independently on each segment.

# Diagnostics & Interpretation

The pipeline outputs a diagnostics CSV with these fields per segment:

- target\_type

- insight\_type

- target\_key (optional)

- segment\_start

- segment\_end

- aic

- fallback (bool)

- change\_point (start index if any)

Use these diagnostics to:

- Identify series where model fitting failed and fallback was used.

- Inspect AIC values to compare model complexity choices per segment.

- Locate change points and inspect pre/post behavior.

# Example Visualization (embedded)

The plot below is an example output for the pair-level series "account behavior". It contains model predictions, prediction intervals, anomalies, and change point lines.

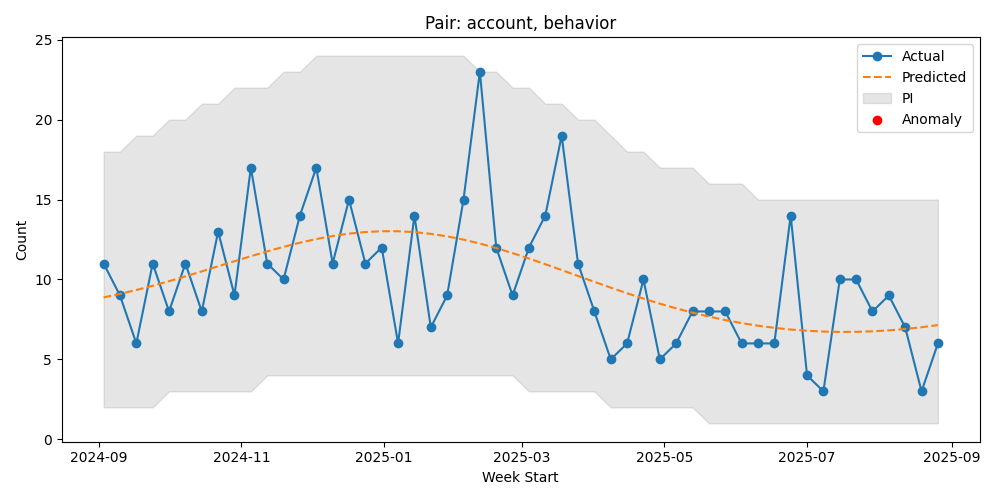


Figure explanation:

- Blue line: Actual weekly counts

- Dashed line: Model-predicted mean

- Gray band: Prediction interval (NB quantiles)

- Red dots: Points flagged as anomalies (outside PI)

- Purple vertical lines: Detected change points where the model was refit

# Diagnostics Sample (first 10 rows)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| target\_type | insight\_type | segment\_start | segment\_end | aic | fallback | change\_point | target\_key |
| account | behavior | 0 | 52 | 277.59967944361887 | False | nan | nan |
| account | behavior | 0 | 52 | 108.4647698839852 | False | nan | ACC-35601 |
| account | behavior | 0 | 49 | 134.9133008604157 | False | nan | ACC-40624 |
| account | behavior | 0 | 52 | 218.59461400621225 | False | nan | ACC-48555 |
| account | behavior | 0 | 52 | 206.33714063455773 | False | nan | ACC-75506 |
| account | behavior | 0 | 51 | 159.37831484343596 | False | nan | ACC-78042 |
| account | behavior | 0 | 50 | 163.12026473098294 | False | nan | ACC-93924 |
| account | opportunity | 0 | 52 | 279.2136305165764 | False | nan | nan |
| account | opportunity | 0 | 52 | 225.884326244132 | False | nan | ACC-10602 |
| account | opportunity | 0 | 51 | 170.53079539790625 | False | nan | ACC-21139 |

# Implementation Notes & Reproducibility

Key files and where to find them:

- copilot\_dev/lightweight\_nb\_anomaly.py - main pipeline

- copilot\_dev/run\_lightweight.py - runner script

- copilot\_dev/requirements.txt - dependencies for enhanced pipeline

Reproducibility checklist:

- Ensure virtualenv is activated and dependencies from requirements.txt are installed.

- Run the runner script to regenerate plots and diagnostics.

# Key Code Snippets

# build\_design: create design matrix with Fourier terms  
P = 52.0  
X = [np.ones(len(g))]  
for k in range(1, K+1):  
 X.append(np.sin(2\*np.pi\*k\*g['t']/P))  
 X.append(np.cos(2\*np.pi\*k\*g['t']/P))  
X.append(g['t']/P)  
X = np.column\_stack(X)

# Gen AI Prompt (for Copilot/ChatGPT)

Use the prompt below to reproduce or adapt this pipeline for a new dataset:

I have weekly count data in CSV format, with columns like `target\_type`, `insight\_type`, `target\_key`, `insight\_date\_time`, and I want to detect anomalies for each (target\_type, insight\_type) pair and key.  
Please develop a robust Python pipeline that:  
- Aggregates the data to weekly counts for each (target\_type, insight\_type, target\_key).  
- For each (target\_type, insight\_type) pair and key:  
 - Fits multiple Negative Binomial (NB2) models with different numbers of Fourier terms (for seasonality) and selects the best model using AIC.  
 - Detects change points in the time series using the `ruptures` library (with a log1p transform and `model='l2'`), and if a change point is found, fits separate models to each segment.  
 - Uses a robust rolling window fallback if model fitting fails.  
 - Calculates prediction intervals using the NB2 model and flags anomalies outside the interval.  
 - Collects diagnostics (AIC, fallback usage, change points) for each segment and saves them to a CSV.  
 - Generates and saves plots for each series, showing actuals, predictions, intervals, anomalies, and change points.  
- Outputs:  
 - Anomaly flags for each key and pair as CSVs.  
 - Diagnostics CSV.  
 - Plots for each series in a diagnostics folder.  
- Use only open-source Python libraries (pandas, numpy, statsmodels, ruptures, matplotlib).  
- The code should be robust, modular, and ready to run on new data with similar structure.  
  
Please provide the full code, including requirements, and ensure it is thoroughly tested and handles edge cases.