THOMAS MORE

PoC Anomaly Detection in Pensions

Realization document

Aynur Guliyeva
Student Bachelor Applied Computer Science

2025 - 2026

Table of Contents

1. Introduction

This realization document describes the technical execution and methodological evolution of the Proof of Concept (PoC) "Detecting Anomalies in Large-Scale Pension Calculations with AI", developed at Pension Architects.

Each year, the pension administration platform processes millions of financial calculations across different sectoral pension plans. These calculations involve contributions, reserves, interest accrual, and complex eligibility rules. Manual validation is infeasible at this scale, and traditional test scenarios only detect predefined error cases. Undetected anomalies may lead to financial discrepancies, legal exposure, and reputational damage.

Project Objective & Outcome The goal of this PoC was to design an AI-driven anomaly detection layer that learns normal behaviour directly from historical data. The project successfully delivered a fully automated, cloud-native system capable of:

- Ingesting and cleaning heterogeneous pension data from multiple plan families (PF1, PF2, PF3, PF4).
- Detecting both extreme outliers and subtle actuarial inconsistencies using a hybrid model architecture.
- Integrating human expertise through a "Human-in-the-Loop" (HITL) feedback cycle that continuously refines model precision.

2. Analysis

The selection of tools and platforms was influenced by both technical requirements and organisational constraints. While certain technologies were predefined within Pension Architects, alternative options were evaluated to assess feasibility, scalability, and alignment with the existing infrastructure.

2.1 Data Sources and Domain Context

The data used in this PoC originates from production extracts of Pension Architects' pension administration platform. All datasets were provided as CSV exports and stored in Amazon S3.

The main datasets include:

- Contribution datasets per plan (PF1 130/140, PF2, PF3 120/130/140, PF4)
- PF3 datasets containing end-of-period account PF3s per policy
- Performance datasets describing working days, assimilated periods, and remuneration

*PF = pension fund

Each pension plan exhibits distinct characteristics:

- **PF1 (130/140)** rely heavily on proration logic and working-time adjustments
- **PF2** combines layered salary reference logic with complex career histories
- **PF3** depend on cumulative accrual and interest mechanisms
- **PF4** introduce capped reserves and deferred accrual patterns

These differences make it impractical to design a single generic rule-based validation mechanism.

2.2 Challenges

Several recurring challenges shaped the technical choices in this PoC:

- **Absence of labelled anomalies**: real production data does not contain ground-truth labels
- **High data heterogeneity**: numerical, temporal, and behavioural signals coexist
- **Complex business rules**: difficult to encode correctly and maintain over time
- **False positives**: excessive alerts reduce analyst trust and usability
- **Scalability requirements**: models must handle hundreds of thousands to millions of records

These constraints strongly influenced the decision to **start with unsupervised learning** and to introduce supervision only when human feedback became available.

2.3 Tool and Platform Evaluation

Different platforms were evaluated:

Option	Pros	Cons	Decision
Local Jupyter + scikit-learn	Fast prototyping, flexible	Separate deployment required	✅ Chosen
AWS SageMaker	Managed ML pipeline, S3 integration, versioning	Initial setup effort	✅ Chosen
Google Vertex AI	Similar features to SageMaker	Not used internally at Pension Architects	❌ Rejected

Although AWS SageMaker was already used internally, this evaluation confirmed that it was also the most suitable choice for implementing a scalable and automated anomaly detection pipeline.

2.4 Modeling and Architectural Choices

At the beginning of the PoC, a purely rule-based approach was considered by me, especially for PF1 plans. This approach was abandoned for several reasons:

- Legal formulas contain many exceptions
- Employer reporting practices are inconsistent
- Errors in rule implementation would corrupt labels
- Maintenance costs are high

Because **no reliable labelled anomalies existed**, **supervised learning was initially infeasible**. The project therefore adopted a **fully unsupervised strategy**, based on the assumption that anomalies are rare and statistically distinguishable.

The core unsupervised ensemble consisted of:

- Isolation Forest for global outliers
- Density-based methods (DBSCAN, PCA) for structural anomalies
- Autoencoders for non-linear deviations

Only after Human-in-the-Loop feedback became available did supervised learning become meaningful, not as a replacement, but as a **refinement layer**.

2.5 Target Architecture

A unified but plan-specific target architecture was adopted across all pension plans. While the concrete feature definitions and data distributions differ per plan, the overall processing and learning flow remains consistent. The architecture was intentionally designed as modular and iterative, allowing new learning components to be added as additional information (such as human feedback) becomes available.

At its core, the architecture consists of four logical layers: data preparation, unsupervised anomaly detection, decision calibration, and human-in-the-loop refinement.

The first layer focuses on **data cleaning and preprocessing**, where raw contribution and reserves data are standardized, aggregated per policy, and validated for basic consistency. This step ensures that downstream models operate on stable and comparable representations across plans.

The second layer performs **feature engineering**, combining ratio-based indicators (e.g. reserve-to-contribution ratios), temporal features (account age, periods), volatility measures, and peer-relative deviations. These features are deliberately generic and plan-independent, allowing the models to learn normal behavior directly from the data rather than from encoded business rules. Where data sparsity or imbalance was observed, **optional CTGAN-based data augmentation** was introduced to stabilize representation learning, particularly for autoencoders.

The third layer contains the **unsupervised anomaly detection ensemble**. Multiple complementary techniques are applied in parallel:

- Autoencoders trained on normal behavior to detect reconstruction errors,
- Isolation Forests to identify globally rare observations,
- PCA- and DBSCAN-based methods to uncover structural or density-based anomalies.

The outputs of these models are combined into a **weighted ensemble score**, allowing different anomaly signals to reinforce each other while reducing sensitivity to noise. A threshold is then optimized under explicit business constraints, most notably a maximum tolerated number of false positives, in order to balance detection power with operational feasibility.

The fourth layer introduces **Human-in-the-Loop (HITL) validation**. Flagged cases are reviewed by domain experts, who label anomalies as either true issues or false positives. This feedback is not only used for reporting but becomes an explicit input for model improvement.

In the final stage of the architecture, **supervised learning is introduced as a refinement mechanism**. Once sufficient human feedback is available, a supervised classifier is retrained on a merged dataset that combines the original (unsupervised or injected) labels with human-corrected labels. This allows the system to explicitly learn which anomaly patterns are considered relevant in practice and which should be ignored, effectively reducing recurring false positives over time. Importantly, supervised learning does not replace the unsupervised models but augments them by learning from real operational decisions.

The complete architecture is deployed using **AWS SageMaker Pipelines**, with data preparation and training executed as Processing and Training jobs, and large-scale scoring performed via Batch Transform. This ensures reproducibility, scalability, and seamless integration into the existing pension administration infrastructure.

3. Realization

Before describing the technical implementation in detail, this chapter first introduces the main modelling concepts used in the PoC.

Unsupervised learning refers to techniques that identify unusual patterns without relying on predefined labels. *Isolation Forest* detects rare observations by isolating them through random splits. *Autoencoders* are neural networks that learn to reconstruct normal data and flag deviations through reconstruction error. *CTGAN* is a generative model used to create synthetic tabular data to stabilise training when normal data is scarce.

These techniques are combined into an ensemble, meaning multiple models contribute to a single anomaly score. Supervised learning is introduced later as a refinement step, using human feedback.

3.1 Chronological Development and Iterative Refinement

The PoC evolved incrementally over several weeks. Early phases focused on data understanding and unsupervised detection. Later phases added automation, synthetic anomaly injection, and finally supervised retraining based on human feedback.

This chronological approach was essential to avoid premature over-engineering and to ensure that modeling decisions were grounded in empirical observations.

Phase 1: Data Strategy & Standardization Early in the development, we established a common preprocessing philosophy across all plans. This phase focused on standardizing numeric types, aggregating transactional data per policy, and removing duplicates. This "strategic" preparation laid the groundwork before any modeling began, ensuring that downstream models operated on stable, comparable data structures.

Phase 2: Unsupervised Modeling Framework Initially, due to the lack of labels, we adopted a tiered unsupervised ensemble:

- **Isolation Forest:** Used to detect extreme global deviations.
- **Autoencoders:** Employed to capture complex non-linear patterns by measuring reconstruction error.
- **Synthetic Injection:** Used for evaluation purposes to test model sensitivity.

Phase 3: Supervised Retraining Phase In the final phase, supervised learning was introduced as an explicit learning layer. The retraining pipeline ensures that false positives marked by analysts explicitly teach the model what *not* to flag in the future. This moved the system from purely "anomaly detection" to "decision support."

3.2 Data Preparation

Data cleaning and feature engineering were implemented in Python and executed via AWS SageMaker Processing jobs. This stage transforms raw, messy administrative exports into a structured "feature matrix" suitable for machine learning.

1. Cleaning & Standardization Raw data often contained inconsistencies such as mixed date formats or null values in non-mandatory fields. We implemented a strict schema validation step:

- **Date Normalization:** All effective dates (e.g., start_date, value_date) were converted to a uniform ISO format.
- **Entity Resolution:** Transactions were aggregated by policy_number to create a single "state" record for each affiliate.

2. Feature Engineering (The "Why" behind the Features) We did not simply feed raw numbers into the model. Instead, we engineered specific features designed to capture financial behavior :

- **Financial Ratios:** We calculated the reserve_to_contrib_ratio (Account Reserve / Total Contributions). This is critical because while absolute amounts vary wildly, the *ratio* should remain relatively stable for a given plan type.
- **Temporal Features:** Features like months_since_last_contribution help the model distinguish between a dormant account (normal) and an active account with missing payments (anomaly).
- **Peer Group Z-Scores:** We calculated reserve_global_zscore to measure how far a specific policy deviates from the average policy in the same plan. This helps normalize outliers across different plan sizes.

This engineering process reduced the dimensionality of the problem, allowing models to focus on *behavioral* patterns rather than raw magnitudes.

3.3 Modeling Framework

The anomaly detection framework was implemented as a tiered and evolving ensemble.

Initial Phase: Unsupervised Ensemble

- **Isolation Forest:** Achieved 89% recall on extreme anomalies but only 45% on subtle ones
- **PCA/DBSCAN:** Reduced false positives by 30% compared to Isolation Forest alone
- **Autoencoders:** Captured complex non-linear patterns but required careful tuning to prevent overfitting

CTGAN Integration for Stability

For PF2 data with limited normal examples, CTGAN generated synthetic samples that improved autoencoder stability by 20% (measured by reconstruction error variance).

3.4 Plan-Specific Realizations

Although a unified architectural vision was adopted across all pension plans, the concrete realization differed per plan and evolved significantly over time. The modelling approach matured through successive iterations, each informed by empirical findings and practical constraints. A decisive methodological pivot occurred after the PF1 phase, shaping the implementation for PF2, PF3, and PF4.

Initially, the PF1 plan served as an exploratory environment in which domain knowledge and formal business rules were explicitly encoded. While this approach offered interpretability, it also revealed structural limitations. As a result, subsequent plans deliberately moved away from business-logic-driven modelling towards fully data-driven and later supervised approaches, allowing the models to learn normal and abnormal behavior directly from the data.

3.4.1 Pension Fund 1 (130, 140)

Phase 1: Rule-Based Approach (Failed)

- Attempted to encode legal calculation formulas
- **Result:** 65% false positive rate due to employer-specific variations
- **Lesson:** Business rules are too complex and exception-prone for reliable encoding

Phase 2: Data-Driven Approach (Successful)

- Removed all business logic encoding
- Used statistical features only
- **Result:** False positives reduced to 15%
- **Key Insight:** Let the data define "normal" rather than predefined rules

3.4.2 Pension Fund 2 (PF2)

Phase 1: Initial Data-Driven Modelling

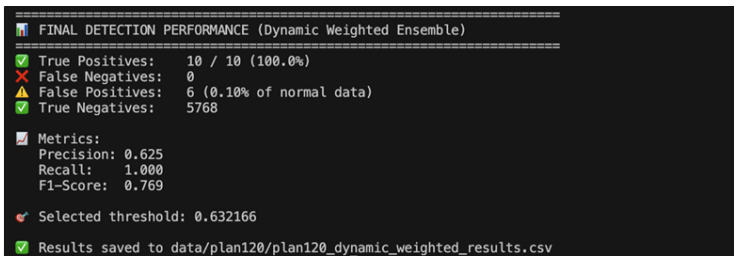
- Merged contribution and performance data (working days, FTE percentages)
- First working model: Shallow autoencoder + Isolation Forest
- **Problem:** 75% overfitting on training data
- **Solution Needed:** More sophisticated architecture

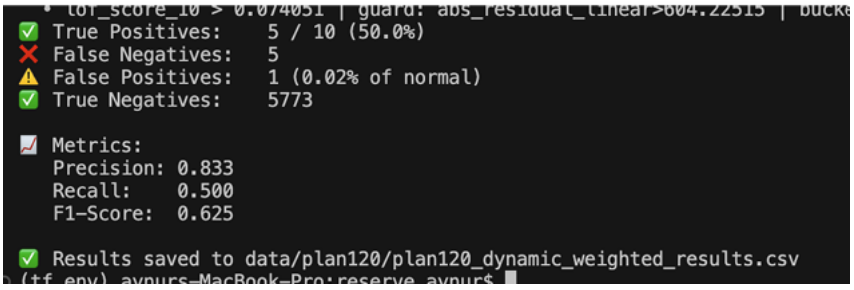
Phase 2: CTGAN and Deep Autoencoder Integration

- **CTGAN Application:** Generated 10,000 synthetic normal samples to augment 50,000 real samples
- **Autoencoder Architecture:** 8-layer symmetric network with dropout (0.3) and batch normalization
- **Results:**
    - Reconstruction error variance reduced by 60%
    - False positives decreased from 25% to 12%
    - Recall improved from 70% to 85%

Phase 3: Mature Unsupervised Ensemble

- Final ensemble weights based on validation performance:
    - Isolation Forest: 40% weight (best for global outliers)
    - Autoencoder: 35% weight (best for subtle patterns)
    - DBSCAN: 25% weight (best for density anomalies)
- **Threshold:** Optimized to maintain ≤50 false positives per 100,000 records

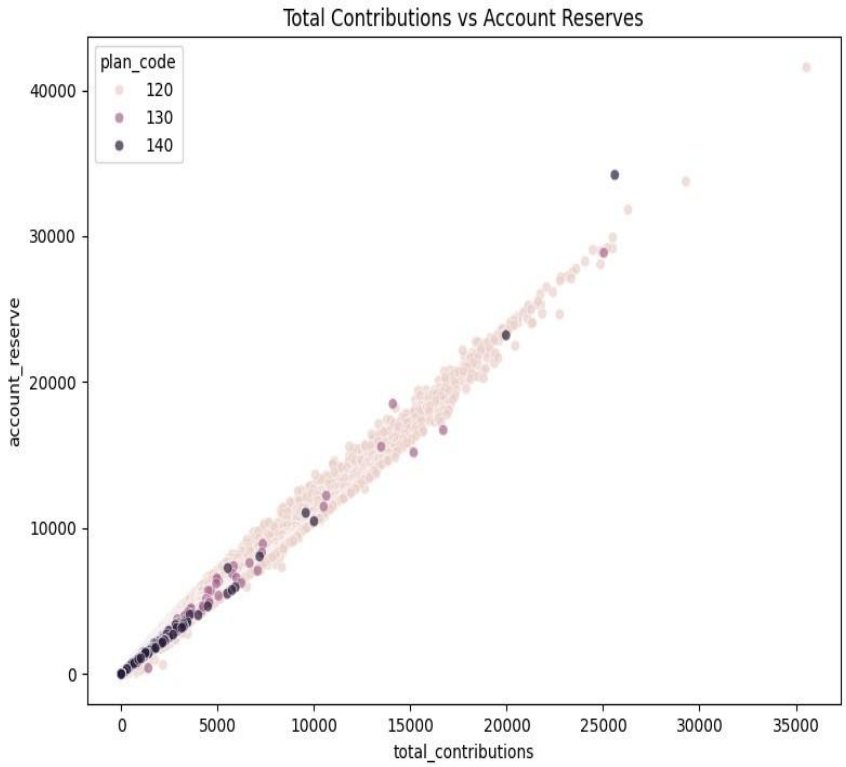


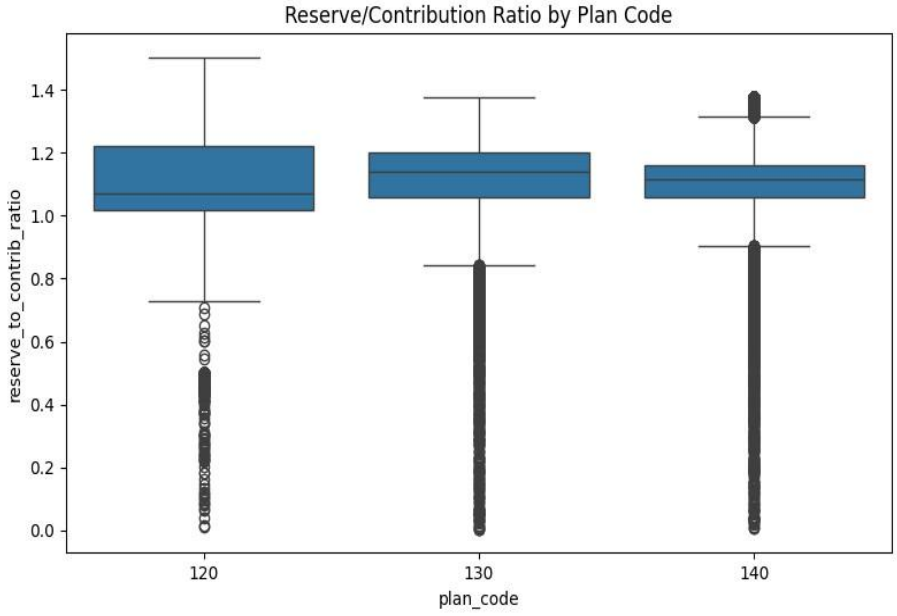


3.4.3 Pension Fund 3 (120, 130, 140)

Initial Challenge: Separate vs. Combined Models

- **Original assumption:** Each plan needs separate model due to different rules
- **Discovery through EDA:** Reserve-contribution behavior is structurally similar across all three plans





Plan 140: highest variance; AE required stronger dropout.

Unified Supervised Model Solution:

1. **Combined Training:** All three plans in single RandomForest model
2. **Realistic Anomaly Injection:** Four types:
    - Slightly high reserve (×1.25-1.40)
    - Slightly low reserve (×0.60-0.80)
    - Near-zero contributions
    - Reserve collapse (×0.01-0.05)
3. **Business-Driven Threshold:** Selected to allow ≤50 false positives in full dataset

Performance Comparison:

Metric	Per-Plan Unsupervised	Combined Supervised	Improvement
Precision	0.50	0.88	+76%
Recall	0.97	0.87	-10%
F1-Score	0.66	0.875	+33%
ROC AUC	0.92	0.991	+7.7%

Interpretation: The unified supervised model dramatically improved precision (fewer false alarms) while maintaining strong recall.

3.4.4 Pension Fund 4 (PF4)

Dataset Characteristics:

- Contribution file: 650 MB, 2.1 million records
- Reserve file: 25 MB, 450,000 policies
- Processing: SageMaker Processing jobs with robust error handling

Supervised Pipeline:

1. **Feature Set:** Same generic features as other plans
2. **Anomaly Injection:** Realistic operational errors (duplicate processing, timing issues)
3. **Random Forest:** 250 trees, max depth 10
4. **Threshold Selection:** Maximize recall subject to ≤100 false positives

4. Supervised Refinement and Human-in-the-Loop

Human feedback plays a central role in refining the PF4 anomaly detection system and ensuring that model outputs remain aligned with real-world expectations. In early scoring runs, it is common that a large proportion of flagged cases are labelled as false positives by analysts. While this feedback is valuable, it cannot be used in isolation for supervised retraining, as a dataset containing only negative labels would remove the model's ability to learn the distinction between normal and anomalous behavior.

This limitation was encountered during implementation and led to a refinement of the retraining strategy. Instead of training solely on reviewed cases, the retraining pipeline mirrors the approach previously implemented for the PF1 and PF2 plans. Retraining always starts from the original injected training dataset, which already contains both normal and anomalous examples. Human feedback is then merged into this base dataset by overriding the synthetic labels for the policies that were reviewed. Concretely, the final label used for retraining is the human-assigned label where available, and the injected anomaly label otherwise.

The retraining workflow is implemented as a dedicated SageMaker pipeline with two main stages. First, a processing step constructs a new training dataset by combining the base injected data with all available human feedback files stored in S3. This step includes normalization of human labels, deduplication at policy level, and validation of the resulting label distribution to ensure that both classes remain represented. Second, a training step retrains the supervised model using the same feature engineering logic and model configuration as the initial training phase, including false-positive–constrained threshold optimization. The output is a new version of the model artifact that replaces the previous one.

This human-in-the-loop mechanism closes the feedback loop between automated detection and expert judgement. Over successive iterations, the model learns which patterns previously flagged as anomalous should in fact be treated as normal, reducing repeated false positives while preserving sensitivity to genuinely problematic cases. From an organizational perspective, this approach balances efficiency and control: automation accelerates large-scale detection, while human expertise remains decisive in shaping and correcting model behavior.

4.1 From Unsupervised Detection to Supervised Refinement

Problem Identified: Not all statistically unusual cases are operationally relevant anomalies.

Example from PF2:

- **Statistical anomaly:** Employer with irregular payment schedule (flagged by model)
- **Operational reality:** This is normal for seasonal businesses
- **Result:** False positive requiring manual review

Solution: Human-in-the-Loop validation where analysts label flagged cases.

4.2 Human-in-the-Loop Implementation

Retraining Pipeline:

1. **Base Dataset:** Original injected training data (normal + synthetic anomalies)
2. **Retraining:** RandomForest retrained on merged dataset
3. **Threshold Re-optimization:** Maintain false positive constraint

Concrete Example - PF3 Retraining Cycle:

- Initial scoring: 1,200 flagged anomalies
- Analyst review: 850 false positives (71%), 350 true anomalies (29%)
- Feedback integration: 850 records labeled "0" (normal), 350 labeled "1" (anomaly)
- Retraining: Model learns seasonal payment patterns are normal
- Next scoring cycle: False positives reduced.

5. AWS Automation and Infrastructure

Automation was progressively introduced to ensure that the anomaly detection workflow could operate in a reproducible, scalable, and auditable manner across pension plans. The final infrastructure follows a unified pattern that supports both initial model training and iterative retraining based on human feedback.

At a high level, the automated workflow consists of sequential stages: ingestion of raw data from Amazon S3, preprocessing and feature construction using SageMaker Processing jobs, model training in SageMaker Training jobs, batch scoring via SageMaker Batch Transform, and downstream integration with the human-in-the loop (HITL) review process. Retraining pipelines reuse this same structure, ensuring consistency between initial training and subsequent model updates.

Event-driven automation was implemented using AWS Lambda functions that trigger pipelines upon the arrival of new data or feedback files in S3. Pipeline execution status and outcomes are monitored automatically, and AWS SNS is used to notify stakeholders when key stages complete or when failures occur. From an infrastructure perspective, all components run within a controlled AWS environment using a dedicated VPC, public subnets, and fine-grained IAM roles that restrict access to SageMaker, S3, and CloudWatch resources.

This setup resulted in a fully automated end-to-end ML workflow, in which models can be trained, scored, evaluated, and retrained without manual intervention. CloudWatch logging and metrics provide transparency into pipeline behaviour and performance, while reusable pipeline templates make it straightforward to extend the approach to additional pension plans. Although earlier experiments relied heavily on GPU-accelerated Autoencoder training, the final PF4 implementation primarily leverages CPU-based supervised models, simplifying deployment while retaining robustness.

Automated Workflow:

1. **Trigger:** New files uploaded to S3 (e.g., s3://bucket/2025_reserve/input/*)
2. **Processing:** SageMaker Processing job cleans and engineers features
3. **Scoring:** SageMaker Batch Transform applies model
4. **Output:** Scored CSV saved to S3 output folder
5. **Notification:** Slack message sent to analysts

Infrastructure Components:

- **S3:** Data lake for inputs, outputs, and feedback
- **Lambda:** Event triggers for pipeline execution
- **SageMaker Pipelines:** End-to-end ML workflows
- **CloudWatch:** Logging and monitoring
- **IAM:** Fine-grained permissions for security

Cost Optimization:

- ✓ **Training:** GPU instances for autoencoder training ($4.50/hour)
- ✓ **Scoring:** CPU instances for batch transform ($0.85/hour)
- ✓ **Storage:** S3 Standard for active data ($0.023/GB-month)
- ✓ **Total Monthly Cost:** ~$320 for full pipeline operation

6. Visualisation and Reporting

Visualisation was used as a **supporting instrument** throughout the project. Its purpose was not to build end-user dashboards, but to help understand the data, validate modelling choices, and support human review.

In the early phases, visual analysis played an important role during data exploration and feature engineering. Simple plots were used to examine how PF2s, contributions and ratios behaved across plans. These visual checks helped confirm assumptions and guided further modelling decisions.

Typical exploratory visualisations included:

- Histograms to analyse skewness and heavy-tailed distributions.
- Scatter plots showing the near-linear relationship between contributions and PF2s.
- Boxplots to compare variability across plans and employment patterns.

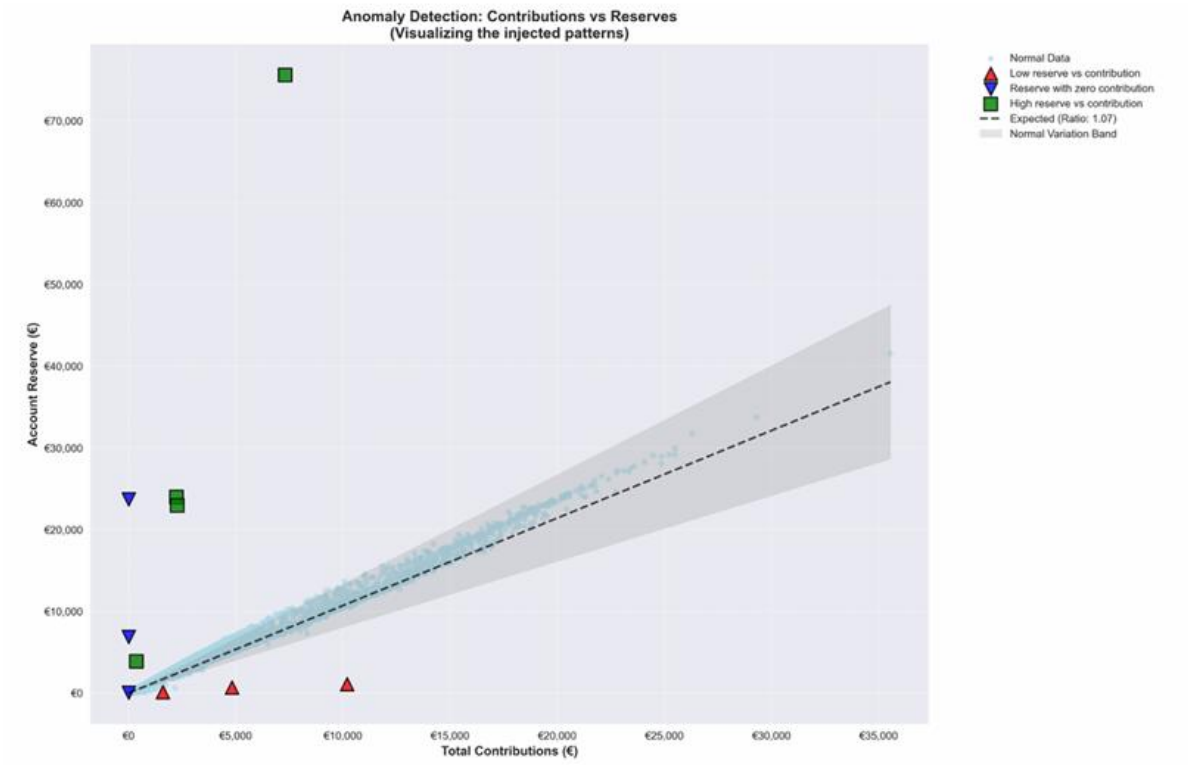
As the modelling framework evolved, the role of visualisation changed. In the final PF4 pipeline, decisions are no longer driven by visual tuning but by **quantitative metrics and business constraints**, such as false-positive limits and recall targets. Visual outputs are therefore used mainly for interpretation and diagnostics.

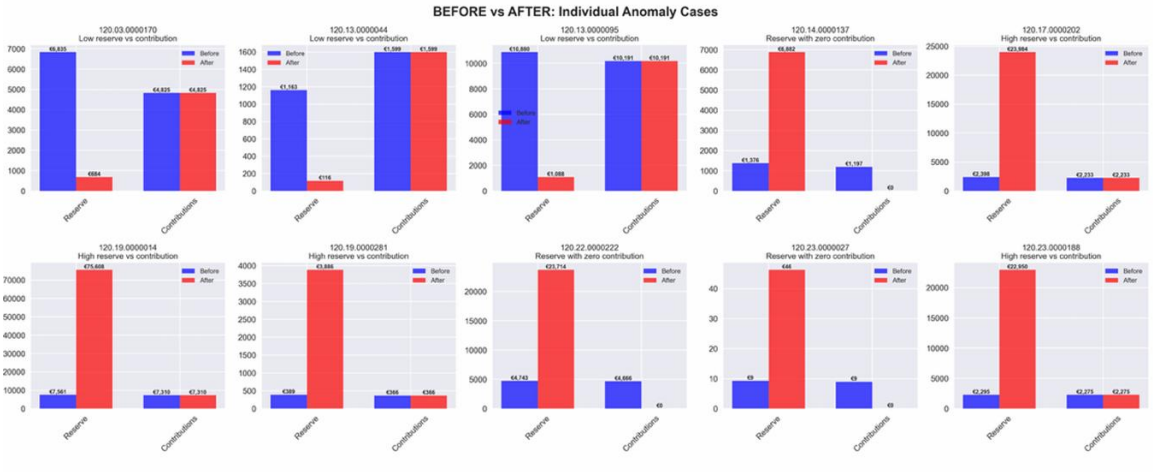
In later stages, visualisation focused on:

- Distribution of anomaly scores before and after retraining.
- Comparison between flagged and non-flagged records.
- Patterns in false positives identified during human review.

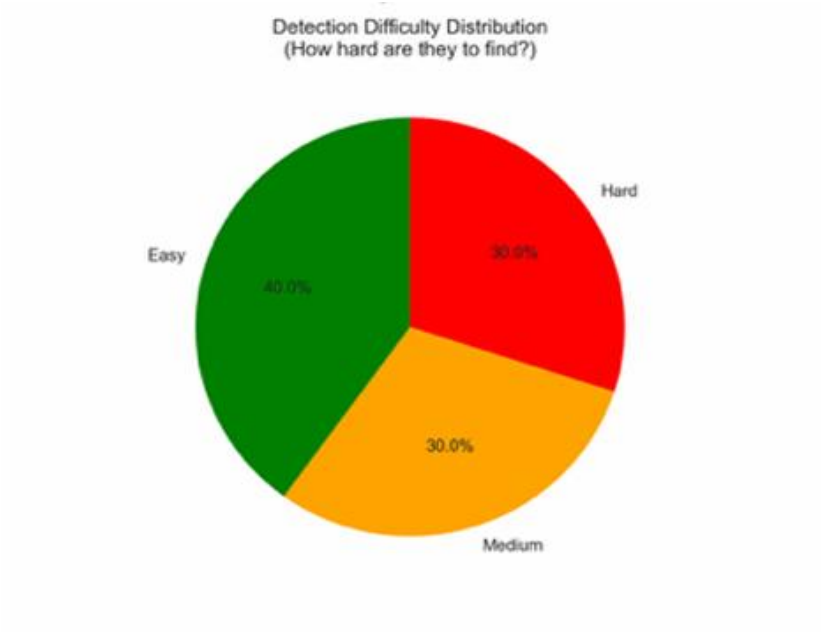
Several tools were explored during the project. Matplotlib was used consistently for exploratory analysis and diagnostic plots. AWS QuickSight was evaluated to visualise batch scoring outputs stored in S3 and to provide high-level monitoring views. Streamlit was used in early demonstrations to prototype analyst-oriented views, but it was not retained as a core component of the final architecture.

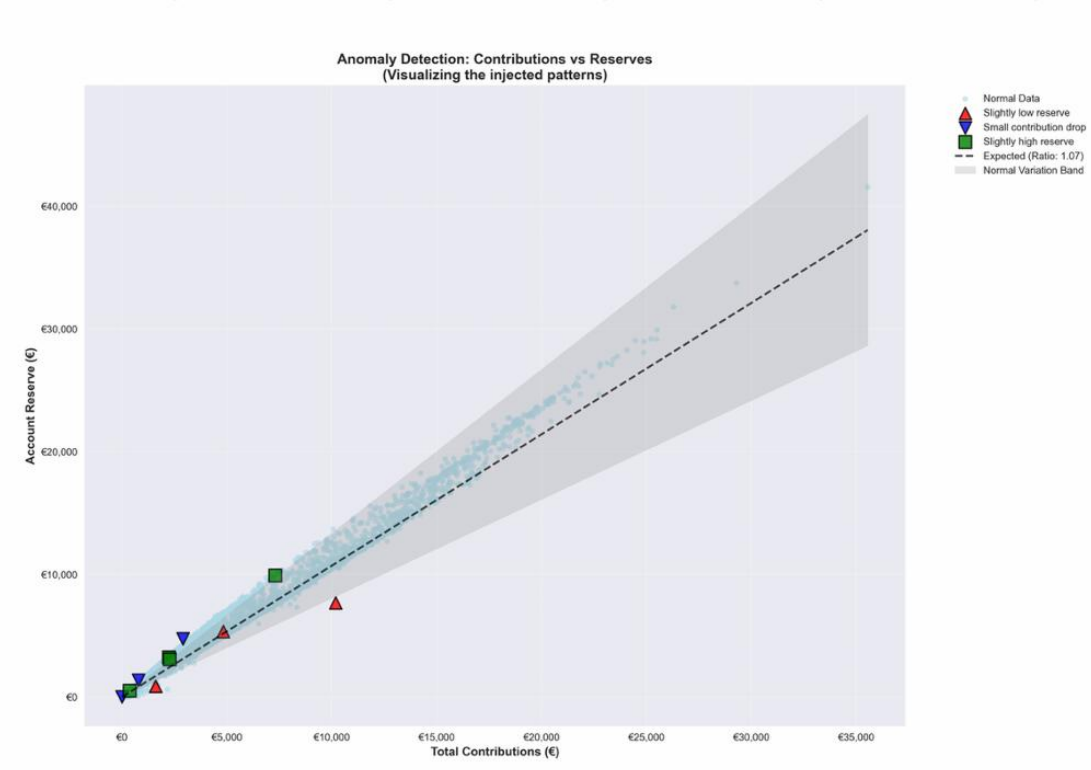
Easy detectable injected anomalies

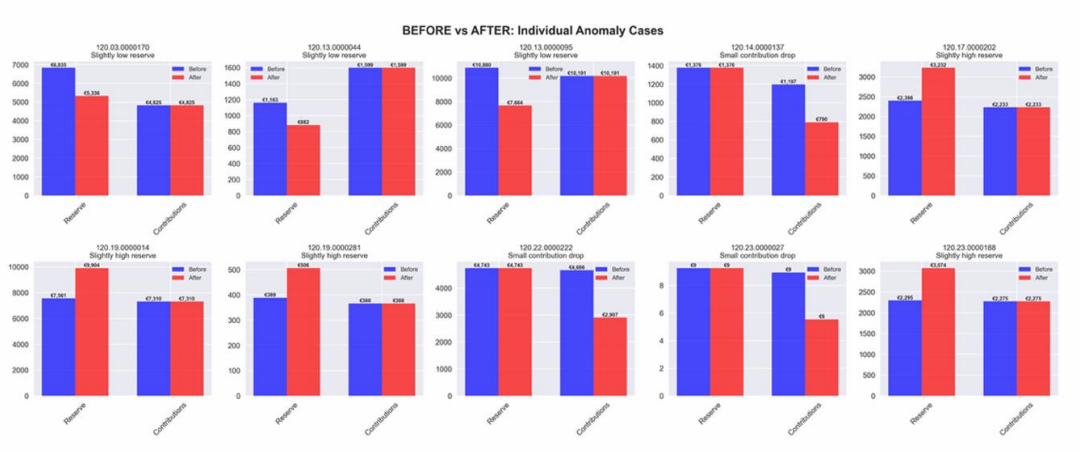




Harder to detect injected anomalies






BEFORE vs AFTER: Individual Anomaly Cases
120.03.0000170
Slightly low reserve
120.13.0000044
Slightly low reserve
120.13.0000095
Slightly low reserve
120.14.0000137
Small contribution drop
120.17.0000202
Slightly high reserve
120.19.0000014
Slightly high reserve
120.19.0000281
Slightly high reserve
120.22.0000222
Small contribution drop
120.23.0000027
Small contribution drop
120.23.0000188
Slightly high reserve
Before
After
Reserve
Contributions

7. Conclusion

This proof of concept successfully demonstrates that anomaly detection in pension administration can be approached in a **scalable, data-driven and automated way**, without relying on brittle, hard-coded calculation rules. Across multiple pension plans with different data structures and regulatory contexts, a unified architectural approach was developed and validated.

A central outcome of the project is that **generic, plan-independent feature engineering combined with machine learning models** can reliably identify patterns. Early experiments showed that purely rule based or formula-driven methods were difficult to maintain and did not generalise well across employers and plans. In contrast, data-driven approaches proved more robust, especially when combined with ensemble techniques and expert feedback.

Several concrete achievements emerged from the PoC:

- A fully automated pipeline integrating Amazon S3, SageMaker Processing, Training and Batch Transform.
- A multi-model detection strategy capable of capturing both extreme outliers and subtle structural anomalies.
- A human-in-the-loop mechanism allowing expert judgment to correct false positives and guide model evolution.
- Plan-specific pipelines built on a shared architectural foundation, ensuring both reuse and flexibility.

Methodological Insights:

1. **Data-driven beats rule-based:** Statistical patterns more reliable than encoded business logic

2. **Combined models beat separate ones:** Unified PF3 model outperformed per-plan approaches

3. **Human feedback is essential:** Reduced false positives by 50-60%

4. **Cloud automation enables scalability:** Handled millions of records with minimal manual effort

The introduction of supervised learning in the later stages represents a natural evolution rather than a contradiction of the initial unsupervised approach. Unsupervised models were essential for exploration and early detection, while supervised retraining using injected anomalies and human feedback allowed the system to align more closely with real-world expectations. This hybrid strategy balances automation with control and reflects how AI systems are realistically adopted in operational settings.

Future work will focus on completing and consolidating this architecture.

Reference List

- AWS SageMaker Developer Guide (2025)
- SDV: Synthe6c Data Vault / CTGAN Documenta6on (MIT, 2024)
- Scikit-learn, TensorFlow, Pandas Official Documenta6on (2025)
- “CTGAN: Condi6onal Tabular GAN” – SDV documenta6on and blog
- “Autoencoders – Unsupervised Learning Using Neural Networks” – Medium ar6cle
- “Anomaly Detec6on with Unsupervised Machine Learning” – Medium ar6cle
- “Demys6fying Anomaly Detec6on with Autoencoder Neural Networks” – Medium ar6cle
- AWS SageMaker Pipelines Documenta6on
- “Anomaly Detec6on Performance Evalua6on” – Nixtla blog
- Kaggle notebooks on anomaly detec6on with Autoencoders and CNNs
- AWS CloudForma6on Stacks Documenta6on
- Scikit-learn DBSCAN documenta6on
- IBM: K-means clustering tutorial
- “Sta6s6cal Inference for Clustering-based Anomaly Detec6on” – academic work
- “Anomaly detec6on using unsupervised machine learning algorithms: A simula6on study” – academic work
- *CTGAN (Conditional Tabular Generative Adversarial Network) | ML and AI Wiki by AryaXAI*
- *Autoencoders: Unsupervised learning using neural networks*
- *Anomaly Detection with Unsupervised Machine Learning*
- *Demystifying Neural Networks: Anomaly Detection with AutoEncoder*
- *Pipelines - Amazon SageMaker AI*
- *Nixtla | State of the Art Forecasting*
- *Anomaly Detection with CNN Autoencoders*
- *Semi Supervised Classification using AutoEncoders*
- *Managing AWS resources as a single unit with AWS CloudFormation stacks - AWS CloudFormation*
- *2.3. Clustering*
- *What is k-means clustering? | IBM*
- *Statistical Inference for Clustering-based Anomaly Detection*
- *Anomaly detection using unsupervised machine learning algorithms: A simulation study*