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.

The objective of this PoC was to design an AI-driven anomaly detection layer that:

- Learns normal behavior directly from historical data
- Detects irregular or suspicious calculation outcomes
- Integrates into an AWS-based production environment
- Supports Human-in-the-Loop (HITL) validation and iterative improvement

An important characteristic of this project is its **iterative methodological evolution**. The PoC started with exploratory, rule-inspired and unsupervised approaches, and gradually evolved into a **hybrid architecture** in which **unsupervised learning remains central**, but **supervised learning is introduced in the final phase** to explicitly learn from human feedback.

This document therefore intentionally describes **both the unsupervised foundations and the supervised refinement**, as each step was necessary to reach a production-ready and business-aligned solution.

2. Analysis

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

AWS SageMaker was selected for its automation capabilities, integration with S3, model registry, and cost-effective batch scoring.

2.4 Modeling and Architectural Choices

At the beginning of the PoC, a purely rule-based approach was considered, 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

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.

Data Preparation

All plans share a common preprocessing philosophy:

- Standardization of numeric types and dates
- Aggregation of transactional data per policy
- Removal of duplicates and invalid records
- Feature engineering focused on ratios, volatility, and temporal behavior

Crucially, **no explicit business formulas were encoded** in the final pipelines. Domain knowledge was used only to guide feature design, not to define “correct” values.

Unsupervised Modeling Framework

Across plans, a tiered ensemble architecture was adopted:

- **Isolation Forest** detects extreme deviations
- **Autoencoders** capture complex non-linear patterns
- **Density-based methods** identify structural outliers

Synthetic anomaly injection was used exclusively for **evaluation and calibration**, never as ground truth.

For PF2 and, to a lesser extent, PF3 and PF4 plans, **CTGAN** was introduced to generate synthetic *normal* samples, stabilizing Autoencoder training in sparse or imbalanced feature spaces.

Supervised Retraining Phase

In the final phase, **supervised learning was introduced**, not as a replacement, but as an **explicit learning layer on top of unsupervised detection**.

The retraining pipeline works as follows:

1. Base training data (with injected anomalies) is loaded
2. Human feedback CSVs are merged by policy number
3. The final label is defined as:
   human label if present, otherwise injected label
4. A supervised RandomForest model is trained
5. A threshold is optimized under strict false-positive constraints
6. The updated model bundle is stored and used for future scoring

This approach ensures that:

- False positives explicitly *teach* the model what not to flag
- Analyst decisions directly influence future behavior
- The system continuously improves with minimal additional effort

Importantly, supervised learning **does not replace** unsupervised detection. It refines it.

3.2 Data Preparation

Data cleaning and feature engineering were implemented in Python and executed on SageMaker Processing:

- Type conversion and date normalisation.
- Removal of duplicates and imputation/clean handling of missing values.
- Feature engineering focused on:
  - reserve_to_contrib_ratio (reserve / contribution).
  - abs_residual_linear and other residuals.
  - z_score_std, reserve_deviation and related measures.
  - Aggregation per policy with sum() and other statistics across all financial metrics.

3.3 Modeling Framework

The anomaly detection framework was implemented as a **tiered and evolving ensemble**, reflecting both the absence of reliable labels in the initial phase and the

progressive introduction of expert knowledge through human feedback. The modeling strategy therefore combines unsupervised learning for discovery with supervised learning for refinement.

In the first phase, anomaly detection relied entirely on **unsupervised models**, as no trustworthy labels were available in historical pension data. A three-tier ensemble was designed, in which each model type captures a different notion of abnormality. Isolation Forest was used to identify extreme global outliers based on recursive partitioning, providing a strong baseline for detecting unusually large or small reserve patterns. In parallel, a combination of PCA and DBSCAN was applied to uncover structural inconsistencies and density anomalies in the reduced feature space, allowing deviations from dominant data clusters to be identified. Finally, an autoencoder was trained on inferred normal behavior to capture more complex non-linear relationships between contributions, reserves, and temporal features, with reconstruction error serving as an anomaly signal.

Several additional methods were explored during experimentation, including Local Outlier Factor, K-Means clustering, One-Class SVM, and SMOTE-based resampling. These techniques were ultimately excluded from the final ensemble due to limited robustness, sensitivity to hyperparameters, or reduced interpretability in the pension context. The final unsupervised ensemble therefore retained Isolation Forest, PCA/DBSCAN, and the autoencoder, as these models provided complementary perspectives on anomalous behavior while remaining operationally explainable.

To improve the stability of the autoencoder—particularly for plans with limited or highly skewed data such as PF2—**synthetic data generation using CTGAN** was introduced. CTGAN was trained on samples inferred as normal by the first-tier Isolation Forest, allowing the autoencoder to learn a smoother representation of normal behavior without amplifying anomalous patterns. This augmentation step was explicitly constrained to support representation learning rather than to simulate anomalies.

In a second phase, once the system was deployed and **Human-in-the-Loop validation became available**, the modeling framework was extended with **supervised learning**. Domain experts reviewed flagged anomalies and labelled them as either true anomalies or false positives. These labels were merged with the original unsupervised or injected labels to create a corrected training dataset. A supervised classifier (Random Forest) was then trained on the same generic feature set as the unsupervised models, allowing the system to explicitly learn which anomaly patterns are considered relevant in practice.

Supervised learning was therefore not used as a replacement for the unsupervised ensemble, but as a **feedback-driven refinement layer**. Its primary role is to reduce recurring false positives and align anomaly scoring with real operational decisions. This hybrid approach ensures that the system remains capable of discovering novel anomaly patterns, while simultaneously learning from expert judgement to improve precision over time.

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 (130, 140)

The PF1 plans formed the experimental foundation of the project and therefore underwent the most extensive methodological evolution. Several modelling strategies were explored in sequence, each revealing important limitations that informed later design choices.

A. Initial business-logic-driven approach

The first modelling attempt aimed to validate contributions by explicitly reproducing the official PF1 calculation logic. Using legal specifications and internal documentation, contribution amounts were recomputed from reported salaries, working days, and contractual work percentages. Deviations between computed and reported contributions were interpreted as potential anomalies and clustered for inspection.

This approach provided strong interpretability but quickly exposed structural weaknesses:

- Contribution rules contained many exceptions and employer-specific variations.
- Reporting of working days and salaries was inconsistent across employers.
- Small implementation errors propagated directly into incorrect anomaly labels.
- The approach required continuous maintenance and was difficult to generalize across plans.

As a result, rule-based modelling was considered unsuitable as a scalable detection strategy.

B. Hybrid transition phase

In a second phase, explicit formula reproduction was partially abandoned. Contributions and performance data were merged, and more descriptive features were extracted, such as:

- total contribution amounts and counts,
- working days per quarter,
- average and volatile remuneration patterns.

Early anomaly indicators were added, including deviation from expected FTE-based contributions and salary volatility measures. Clustering techniques were applied to identify unusual cases.

Despite improved flexibility, this hybrid approach still suffered from:

- missing or incomplete performance data,
- deviations reflecting employer reporting habits rather than true anomalies,
- instability when combining numerical and categorical information.

This phase confirmed that retaining encoded business expectations continued to bias the models.

C. Fully data-driven unsupervised approach

The next phase marked a decisive shift towards purely data-driven modelling. All formula-based expectations were removed, and only statistical and behavioural features derived from raw data were retained. The guiding principle was to let the models learn empirical patterns of normal behaviour without predefined notions of correctness.

Key characteristics of this phase included:

- contribution intensity features (sum, count, volatility),
- remuneration distributions and temporal patterns,
- removal of any hard-coded actuarial assumptions.

Multiple unsupervised techniques were evaluated, including Isolation Forest, Local Outlier Factor, DBSCAN, PCA-based density checks, and early autoencoder prototypes. This approach already outperformed earlier rule-based and hybrid methods in terms of stability and coverage.

D. Mature pre-cloud pipeline

The data-driven approach was subsequently consolidated into a stable pipeline consisting of structured cleaning, synthetic anomaly injection, and model training. Cleaning steps standardized contribution and performance datasets and produced one record per policy-quarter. Synthetic anomalies were injected to simulate realistic error patterns, with contamination rates tuned around one percent. Training relied primarily on an ensemble of Isolation Forest and autoencoder models, optionally complemented by DBSCAN.

This version represented the final stable unsupervised solution before cloud deployment.

3.4.2 From Unsupervised Detection to Supervised Refinement

While the mature unsupervised pipeline proved effective at surfacing suspicious cases, its deployment exposed an important limitation: not all statistically unusual cases are operationally relevant anomalies. Many flagged records reflected legitimate employer practices rather than errors. This insight led to the introduction of **Human-in-the-Loop validation**, in which analysts reviewed flagged cases and explicitly labelled them as true anomalies or false positives.

These human labels fundamentally changed the modelling landscape. Instead of discarding unsupervised methods, the project incorporated supervised learning as a refinement layer. Labeled feedback was merged with the original injected or inferred labels, allowing a supervised classifier to learn which anomaly patterns were meaningful in practice. This approach preserved the discovery power of unsupervised models while enabling the system to reduce recurring false positives over time.

3.4.3 Generalisation to Pension Funds

Lessons learned from the PF1 evolution directly informed the implementation of the PF2, PF3, and PF4. These plans skipped rule-based modelling entirely and were implemented from the outset using data-driven feature engineering, synthetic anomaly injection, unsupervised detection, and Human-in-the-Loop feedback. In the final iterations, supervised retraining became an integral part of the pipeline, allowing plan-specific behavior to be learned without hard-coding business rules.

This progression—from rule-based experimentation, through unsupervised discovery, to supervised refinement—represents a deliberate methodological choice rather than a contradiction. Unsupervised models provided coverage and novelty detection, while supervised learning aligned the system with expert judgement and operational constraints.

3.4.4 PF2 Plan

The PF2 plan represented one of the most complex parts of the proof of concept. It combines multiple heterogeneous data sources, layered salary construction logic, and strong employer-specific patterns. As a result, it required more advanced feature engineering, augmentation, and ensemble modelling than the other plans.

Phase 1 — Initial data-driven modelling

The first PF2 pipeline was built using raw contribution and performance data, with the goal of creating a functioning anomaly detection flow without relying on explicit business rules. Two main datasets were merged: performance data (working days, FTE percentages, sickness and career breaks) and contribution data (quarterly salaries used to derive the annual reference salary).

Early experimentation quickly revealed that PF2 logic could not be expressed as a direct formula. Quarterly salaries needed to be recombined across time, proration depended heavily on employment transitions, and employer reporting practices varied significantly. Encoding this logic explicitly proved impractical and fragile.

The first working model combined a shallow autoencoder with an Isolation Forest and manually weighted features. While this Tier-3 prototype demonstrated feasibility, it suffered from rapid autoencoder overfitting, limited normal data availability, and simplistic ensemble logic. Density-based methods such as DBSCAN were also tested but provided little benefit due to inconsistent scaling. This phase concluded that a more sophisticated architecture was required.

Phase 2 — CTGAN and deep autoencoder integration

A major turning point for PF2 was the introduction of CTGAN-based synthetic data generation. PF2 data exhibited strong structural regularities but contained relatively few stable “normal” cases. CTGAN was therefore used to generate synthetic normal samples, fill rare feature combinations, and smooth the training distribution for the autoencoder.

Empirical evaluation showed that CTGAN significantly reduced false positives and produced more stable reconstruction error distributions. In parallel, the autoencoder architecture was redesigned into a deeper, symmetric network with batch normalization, dropout, and Gaussian noise injection. This largely eliminated overfitting and improved the separation between normal and anomalous cases.

An ensemble approach emerged during this phase, combining:

- Isolation Forest scores,
- autoencoder reconstruction errors,
- and exploratory DBSCAN signals,
  merged into a weighted anomaly score.

Although substantially more stable, the ensemble still exhibited sensitivity to noise and required further refinement.

Phase 3 — Mature unsupervised ensemble pipeline

The next phase focused on consolidation and robustness. A definitive preprocessing pipeline standardised all fields, merged contribution and performance data

consistently, and generated stable descriptive features capturing working-day intensity, salary patterns, contribution density, volatility, and generic proration-like ratios without hard-coded rules.

CTGAN was further refined through improved training schedules, conditional generation for categorical fields, regularisation adjustments, and early stopping. The resulting synthetic datasets proved reliable across employers.

The final unsupervised ensemble consisted of:

- Isolation Forest for global outliers,
- a deep autoencoder for non-linear deviations,
- PCA and DBSCAN for structural anomalies,
- and a weighted aggregation of all signals.

This configuration aligned closely with the mature architectures later used for PF3 and PF4.

Phase 4 — Human-in-the-Loop and semi-supervised extension

As model maturity increased, human-in-the-loop (HITL) review became an integral component. Analysts reviewed flagged cases and labelled them as true or false anomalies. These labels were then used to retrain supervised classifiers on top of the unsupervised scores.

This phase provided two key benefits: real-world validation of anomaly relevance and a mechanism to reduce false positives by learning from confirmed false alarms. Importantly, supervised learning was introduced only after the unsupervised models had stabilized and served as a refinement layer rather than a replacement.

Phase 5 — AWS integration

The final phase focused on operationalisation. Preprocessing, augmentation, training, scoring, and retraining steps were progressively converted into AWS SageMaker pipelines. Batch Transform jobs were used for inference, S3 events triggered retraining flows, and CloudWatch captured logs and metrics. Human feedback was stored centrally and fed back into the retraining pipeline, enabling continuous improvement.

Summary of PF2 Evolution

Phase	Approach	Key contributions	Limitations

1	Basic AE + IF	First working pipeline	Overfitting, no augmentation
2	CTGAN + deep AE	Major stability gains	Feature consistency issues
3	Full ensemble	Structural + density signals	Complex tuning
4	HITL + supervised layer	Real feedback loop	AWS deployment
5	AWS pipeline	Scalable MLOps	Deployed

3.4.5 PF3 Plans (120, 130, 140)

The PF3 plans form a central pillar of the proof of concept, as PF3 calculations are actuarially critical and directly impact pension entitlements. For plans 120, 130, and 140, the machine learning system focused on the relationship between accumulated reserves, historical contributions, account duration, and employment status. These plans provided an ideal testbed for anomaly detection due to their strong numerical structure and relatively stable behavior in normal cases.

Phase 1 — Exploratory analysis and ratio-based foundations

Initial exploratory analysis aimed to understand how PF3s evolve in relation to contributions, interest accrual, demographic status, and plan-specific variability. Across all three plans, reserves exhibited a broadly linear relationship with cumulative contributions. Normal cases clustered tightly around stable reserve ratios, while extreme deviations were already visible prior to applying any machine learning techniques.

Variance differed significantly across plans: Plan 120 showed the most stable patterns, Plan 130 moderate dispersion, and Plan 140 the highest variance due to mixed employment trajectories and fragmented contribution histories. This early analysis confirmed that reserve-based ratios were highly informative and well suited for anomaly detection.

From this phase, a first set of descriptive features was established, centred on reserve-to-contribution ratios, reserve growth per year, contribution density, and activity indicators. These features formed the basis of the cleaned PF3 datasets.

Phase 2 — Baseline unsupervised modelling

The first modelling attempts combined Isolation Forests and shallow autoencoders to detect anomalous reserve patterns. Isolation Forests proved effective at identifying extreme global outliers but struggled with contextual anomalies inside dense regions.

The autoencoder models, while capable of capturing non-linear relationships, overfit quickly and produced unstable thresholds, particularly when applied across different plans.

This phase demonstrated that simple unsupervised models were insufficient on their own and highlighted the need for more expressive feature engineering and more robust ensemble strategies.

Phase 3 — Feature engineering expansion

To improve signal clarity, the feature set was significantly expanded. In addition to basic financial ratios, temporal features such as account age and time since first contribution were introduced. Peer comparison metrics were added by computing deviations from plan-specific medians and global z-scores, along with percentile-based indicators. Interaction features combining reserve magnitude, contribution intensity, and time further enhanced the model's ability to detect subtle inconsistencies.

These enhancements materially improved model performance and reduced noise, particularly for plans with higher internal variability such as Plan 140.

Phase 4 — CTGAN evaluation

Given the rarity of true anomalies in PF3 data, CTGAN-based synthetic augmentation was evaluated. Experiments showed that CTGAN slightly improved autoencoder generalisation and smoothed latent representations. However, the effect was less pronounced than for PF2, as PF3 data already contained a large and stable population of normal cases.

As a result, CTGAN was retained as an optional component for PF3 plans rather than a strict dependency.

Phase 5 — Ensemble stabilisation

The mature PF3 architecture adopted an ensemble approach combining multiple complementary signals: Isolation Forests for global outliers, autoencoder reconstruction error for non-linear deviations, and PCA/DBSCAN-based density checks for structural anomalies. These signals were merged into a weighted composite anomaly score.

This ensemble addressed the weaknesses of individual models: Isolation Forests alone missed contextual anomalies, autoencoders alone produced false positives at distribution edges, and density-based methods added sensitivity within dense clusters. The resulting configuration aligned closely with the architectures later used for PF2 and PF4.

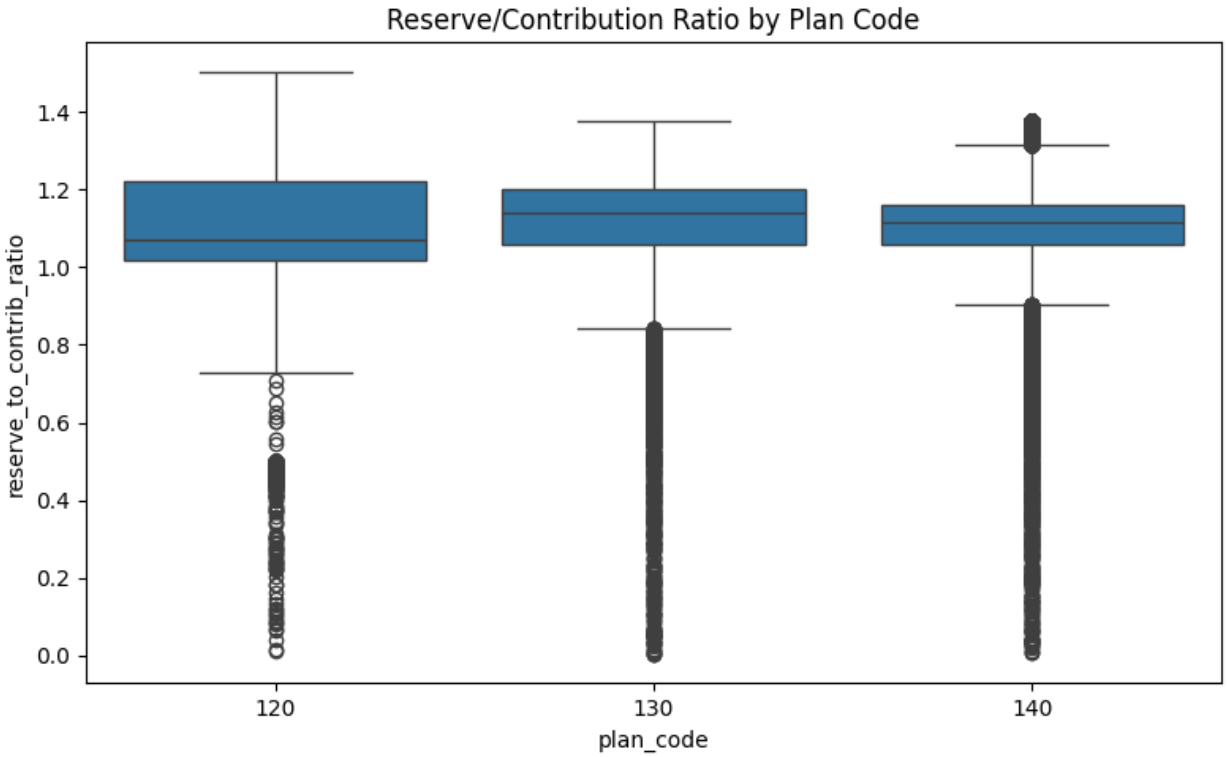
Phase 6 — Plan-specific pipelines

To prevent cross-plan contamination, each PF3 plan was trained and deployed through its own pipeline. While the architectural pattern remained identical, plan-specific distributions required separate models and thresholds.

Plan 120 exhibited the most stable behaviour with few outlier clusters, Plan 130 showed moderate variance, and Plan 140 required the most careful threshold tuning due to heterogeneous employment patterns.

Positioning of supervised learning

As with the other plans, supervised learning was introduced only after the unsupervised PF3 models had stabilized. Human-in-the-loop feedback on flagged PF3 anomalies was later used to refine thresholds and retrain supervised classifiers, primarily to reduce false positives rather than to replace the unsupervised detection logic.



Plan 140: highest variance; AE required stronger dropout.

Threshold optimisation and operationalisation

As the PF3 models matured, threshold selection emerged as a critical factor for practical usability. Raw anomaly scores from individual models were insufficient on their own, as small threshold shifts could lead to large changes in false positive volume.

Early experiments therefore focused on systematic threshold optimization across multiple anomaly scores. A beam-search-inspired approach was explored to evaluate combinations of thresholds across ensemble components, explicitly balancing precision and recall. This allowed the identification of threshold configurations that maintained very low false positive rates while preserving sensitivity to true anomalies. The approach proved particularly useful for Tier-3 alert queues, where analyst workload constraints are strict.

However, this optimization strategy was computationally heavy and difficult to maintain manually. As human feedback became available through the HITL process, threshold optimization was gradually absorbed into supervised retraining. Instead of searching thresholds independently, supervised models learned decision boundaries directly from confirmed false positives and true anomalies, achieving the same objective with greater stability and interpretability.

In parallel, the PF3 pipeline was progressively prepared for operational deployment. Cleaning, feature engineering, training, scoring, and retraining steps were designed

to run as isolated components, enabling migration to AWS SageMaker. Large-scale scoring was executed via Batch Transform, outputs were stored in S3, and flagged cases were routed into the human review cycle. Labeled PF3 anomalies were then reused for supervised fine-tuning, closing the feedback loop.

This evolution transformed threshold optimization from a static tuning exercise into a dynamic, data-driven process embedded in the retraining pipeline.

Summary of PF3 plan evolution

Phase	Approach	Key contribution	Limitation
1	Exploratory analysis	Identification of PF3 ratios	No predictive capability
2	Baseline IF/AE	First anomaly detection pipeline	Unstable thresholds
3	Feature expansion	Strong descriptive signal	Higher dimensionality
4	CTGAN evaluation	Optional robustness gains	Limited added value
5	Ensemble modelling	Complementary anomaly signals	Increased complexity
6	Plan-specific pipelines	Distribution-aware models	Multiple pipelines required
7	Threshold optimisation	FP-controlled alerting	Computational overhead
8	HITL + AWS pipeline	Scalable, adaptive system	Ongoing refinement

3.4.6 PF4 Plan

The PF4 plan follows the same architectural philosophy as the PF2 and PF1 plans, but its implementation required additional technical iteration due to the scale and structure of the underlying datasets. The objective was to construct a generic and

reusable anomaly detection pipeline capable of identifying irregularities in contribution and PF3 data, while remaining independent of plan-specific business logic. Over the course of the project, the PF4 pipeline evolved from an unsupervised exploration phase toward a supervised learning approach augmented by human-in-the-loop retraining.

The pipeline starts from two raw input datasets: a large contribution file (approximately 650 MB) and a smaller PF3 file (approximately 25 MB), both stored in Amazon S3. These datasets are processed using SageMaker Processing jobs. During early implementations, several data ingestion challenges emerged, including inconsistent separators, malformed rows, and the fact that SageMaker mounts input data as directories rather than direct file paths. These issues required explicit handling in the cleaning scripts, such as dynamic resolution of input paths and the use of pandas' Python parsing engine to ensure robustness against corrupted records. Addressing these ingestion constraints was a necessary step to guarantee that the pipeline could operate reliably in a fully automated environment.

After ingestion, a minimal but consistent cleaning and aggregation logic is applied. Contribution data is aggregated per policy to compute generic statistical indicators such as total contribution amount, number of records, average contribution size, and volatility. PF3 data is cleaned separately and then merged with the aggregated contributions using policy identifiers. From the merged dataset, a set of plan-independent features is derived, including temporal characteristics (account age, year, month, quarter), status indicators (active versus inactive), and ratio-based features linking reserves to historical contributions. The guiding principle throughout this phase was to avoid embedding explicit business rules, while still capturing behavioural and structural signals that are meaningful across pension plans.

Initial experiments with PF4 followed the same unsupervised paradigm as PF2 and PF3, relying on Isolation Forests, Autoencoders, and density-based methods. While these models were able to identify extreme outliers, they proved less effective at distinguishing subtle but operationally relevant anomalies, and threshold stability remained a challenge. In addition, the absence of reliable historical labels made it difficult to objectively evaluate model performance beyond qualitative inspection.

To address these limitations, a supervised learning strategy was introduced using synthetically injected anomalies. Instead of random perturbations, the injected anomalies were designed to mimic realistic operational or processing errors, such as missing interest accruals, duplicated contribution processing, incorrect timing of PF3 recognition, or systematic under- or over-estimation of PF3s. Each injected anomaly was explicitly labeled, and original values were preserved for traceability. This approach allowed the construction of a supervised training dataset that reflects

scenarios domain experts would plausibly encounter, while still avoiding reliance on hard-coded business logic.

On this injected dataset, a Random Forest classifier was trained using the generic feature set. Rather than relying on a fixed probability threshold, the training process includes an explicit threshold selection step under a strict false-positive constraint. This design choice reflects an operational reality: in pension administration, each false positive leads to manual review effort. The selected threshold therefore maximizes recall subject to an upper bound on false positives, prioritizing precision while maintaining meaningful anomaly detection capability. The resulting model artifact bundles the trained classifier, the selected threshold, and the list of feature columns, ensuring consistency between training and scoring.

Once trained, the model is applied in a scoring step to new or updated PF4 datasets. Each record receives an anomaly score and a binary prediction based on the learned threshold. The scored output is intentionally not treated as a fully automated decision layer, but as an input to expert review. Analysts inspect flagged cases and assign human labels indicating whether a detected anomaly represents a true issue or a false positive. These human labels are then reused in a retraining pipeline, allowing the supervised model to gradually correct its own false-positive patterns and adapt to real-world data characteristics over time.

This final configuration positions the PF4 pipeline as a mature hybrid system: unsupervised methods informed early exploration and feature validation, while supervised learning with synthetic anomalies and human feedback provides the robustness, evaluability, and operational control required for deployment in a production pension environment.

3.5 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.

3.6 Human-in-the-Loop (HITL)

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.

3.7 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.

In the deployed solution, reporting is intentionally lightweight. Outputs are stored as structured datasets rather than interactive dashboards, so they can be reviewed, audited and reused in downstream processes. Visualization therefore remains an **enabler**, supporting understanding and human-in-the-loop feedback, rather than a primary system objective.

4. 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 irregular contribution and PF2 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.

From a methodological perspective, the project yielded important insights. First, **independent anomaly detection without embedded business logic** leads to more resilient models in complex regulatory environments. Second, ensemble methods consistently reduced false positives compared to single detectors. Third, synthetic data generation using CTGAN proved valuable in contexts where clean normal data are scarce, particularly for structurally complex plans such as PF2. Finally, operational constraints, such as limiting manual review effort, must be explicitly incorporated into model evaluation and threshold selection.

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

- PF1 Plan Model Training Documentation – Aynur Guliyeva (2025, Confluence)
- PF2 Plan Documentation – Aynur Guliyeva (2025, Confluence)
- PF4 Plan v1 Documentation – Aynur Guliyeva (2025, Confluence)
- AWS SageMaker Developer Guide (2025)
- SDV: Synthetic Data Vault / CTGAN Documentation (MIT, 2024)
- Scikit-learn, TensorFlow, Pandas Official Documentation (2025)
- "CTGAN: Conditional Tabular GAN" – SDV documentation and blog
- "Autoencoders – Unsupervised Learning Using Neural Networks" – Medium article
- "Anomaly Detection with Unsupervised Machine Learning" – Medium article
- "Demystifying Anomaly Detection with Autoencoder Neural Networks" – Medium article
- AWS SageMaker Pipelines Documentation
- "Anomaly Detection Performance Evaluation" – Nixtla blog
- Kaggle notebooks on anomaly detection with Autoencoders and CNNs
- AWS CloudFormation Stacks Documentation
- Scikit-learn DBSCAN documentation
- IBM: K-means clustering tutorial
- "Statistical Inference for Clustering-based Anomaly Detection" – academic work
- "Anomaly detection using unsupervised machine learning algorithms: A simulation 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*

Attachements