Technical Deep Dive: Northhaven Analytics Institution-Specific Generative Model (SDG)

Part I: Core Machine Learning Architecture and Data Fidelity

Northhaven Analytics' core offering is a proprietary, dedicated **Institution-Specific Synthetic Data Generator (SDG) ML Model**. This ML architecture is engineered to replicate high-dimensional, highly correlated financial datasets while maintaining strict statistical fidelity and temporal integrity. The system operates entirely independent of real customer data, relying solely on aggregated statistical distributions, metadata, and schema specifications during training. **Our deliverable product to the client is this custom-trained ML artifact, not merely the datasets it generates.**

1. Generative Engine Architecture (The Dedicated ML Model)

The generative engine is built upon a modified **Conditional Generative Adversarial Network (C-CTGAN)** framework, augmented with custom components for managing temporal dependencies and complex dependency structures typical of financial portfolios.

1.1 Structure of the Generator (G) and Discriminator (D)

- **Generator (G):** The Generator maps random noise (latent space) to synthetic data points that match the real data distribution. It uses fully connected layers for static variables and incorporates **Temporal Convolutional Networks (T-CNNs)** and gated recurrent units for time-series data (e.g., transaction sequences, balances over time) to capture long-range dependencies and autocorrelation functions (ACF).
- **Discriminator (D):** The Discriminator is a deep neural network trained to differentiate between real aggregated data distributions (input) and synthetic outputs. Its loss function is reinforced with **probabilistic constraints** derived from domain knowledge (e.g., ensuring Probability of Default (PD) adheres to a 0-1 range; credit utilization cannot exceed the total limit), providing a mechanism for **financial logic embedding** directly within the adversarial feedback loop.

1.2 Dependency Modeling and Probabilistic Framework

The system explicitly models the complex **dependency graphs** inherent in financial data. Before training, a probabilistic model identifies the hierarchical and conditional relationships between features (e.g., 'Client Segment' conditionally determines 'Account Type', which in turn influences 'Transaction Frequency'). This structure is used to condition the Generator,

ensuring that newly synthesized data respects the observed causal flow and multivariate correlations, a significant improvement over standard CTGAN implementations.

1.3 Convolutional Layers and Temporal Sequences

For transactional and behavioral data (e.g., 12-month balance history), the Generator utilizes **1D Convolutional Layers** to efficiently extract features from sequential data. This allows the model to learn short-term temporal patterns (e.g., weekly payment cycles) and maintain consistency across generated time steps.

Part II: Modular Python Library Design and Engineering

The Northhaven platform is implemented as a cohesive, modular Python library designed for robust orchestration, version control, and rapid adaptation across diverse client environments.

2. Modular Framework Components

The entire solution is managed by a central main_controller module, which orchestrates four principal sub-modules:

Module	Functionality	Technical Role
main_controller	Orchestrates the end-to-end pipeline (Training to Generating to Versioning to Deployment). Manages configuration files and initiates processes.	Orchestration Layer, Configuration Management
data_manager	Handles schema ingestion, data serialization, statistical aggregation, metadata management, and output structuring.	Data Engineering, Schema Validation, Statistical Summary Generation
model	Encapsulates the core generative architecture, manages training parameters, executes the adversarial process, and handles new dataset generation commands.	ML Core, Training & Inference Engine

	This is our dedicated ML artifact.	
git_controller	Manages model artifact versioning (e.g., checkpoints, trained weights), statistical input documentation, and synthetic dataset manifests using Git/GitHub.	VCS Integration, Reproducibility Engine

2.1 The git_controller as the Reproducibility Backbone

A critical engineering differentiator is the use of **Git/GitHub** as the backbone for reproducibility and auditability. The git_controller automatically commits:

1. The final trained model weights (the **dedicated ML model artifact**).
2. The metadata and statistical summaries used for training.
3. The schema and constraint definitions.

This guarantees that any synthetic dataset generated can be precisely reproduced, an indispensable requirement for regulatory submissions (e.g., demonstrating consistency for SR 11-7). By versioning the model and metadata, the system ensures *unlimited* storage capacity for historical model states, decoupling storage from the generative process.

Part III: Pipeline, Performance, and Security Isolation

3. End-to-End Pipeline

The operational flow, managed by the main_controller, follows a strict sequence:

1. Training: The data_manager aggregates client-specific statistical inputs. The model module executes C-CTGAN/TSM training until fidelity metrics (e.g., Kullback-Leibler Divergence, correlation retention) converge within a predefined threshold.
2. Generating: A command specifies the required volume (e.g., 500 million records) and conditioning variables (e.g., target 'High Default Risk' cohort). The model executes inference, and the data_manager structures the output.
3. Validation: An internal fidelity check ensures the newly generated dataset maintains the required correlation structures and distributional alignment.
4. Versioning & Deployment: The git_controller tags and commits the newly trained model artifact. The synthetic dataset is then deployed to the client’s designated secure sandbox or API endpoint.

3.1 Continuous-Learning Mechanism (CL)

The platform implements a controlled **continuous-learning (CL) loop**, designed to mitigate Model Drift and enhance fidelity to rare events. After validation, generated synthetic datasets are periodically fed back into the adversarial training process, acting as challenging examples for the Discriminator. This forces the Generator to improve its realism, particularly for complex, multi-variable financial interactions, without ever exposing the original real data. This CL mechanism ensures the **dedicated generative model** remains statistically solid as the client's production environment evolves.

3.2 Security and Client Isolation

Security is enforced at the architectural level through strict **isolation principles**:

- **Dedicated Generative ML Models:** Each client institution receives a physically and logically dedicated instance of the **SDG model**. Models are trained only on the specific institution's statistical footprint and cannot be cross-pollinated or shared.
- **Isolated Dataset Logic:** The data_manager ensures that all schema definitions, constraints, and aggregation logic remain proprietary and isolated within the client's environment definition.
- **Zero PII Input:** The system operates strictly on statistical and structural metadata, never requiring or processing raw PII, thereby establishing a fundamental defense against privacy leakage (GDPR compliance guarantee).

3.3 Performance and Scalability Benchmarks (Market Leader)

The platform is optimized for high-volume financial data generation. Performance benchmarks demonstrate market-leading sub-linear scaling capabilities:

Output Volume	Generation Time (Approx.)	Schema Complexity
1 Million Rows	approximately 8 minutes	Standard (10 features, 1 sequence)
10 Million Rows	approximately 15 minutes	Standard
1 Billion Rows	approximately 16 hours	High (50 features, multiple sequences)

This performance profile allows institutions to rapidly execute Monte Carlo simulations and train deep learning models that require vast quantities of data, transforming development cycles from weeks into hours.

3.4 Example Dataset Structure

The system handles complex hierarchical and temporal structures, such as the following

representation of a consumer debt portfolio:

Layer	Entity	Key Attributes Generated
Static	Client	Demographics, Synthetic Credit Score, Income Distribution
Static	Account	Account Type, Origination Date, Initial Limit, Final Status (Default/Non-Default)
Temporal	Transaction	High-frequency deposit/withdrawal events, category (e.g., payroll, bill payment)
Temporal	Balance	Monthly closing balance, Utilization Rate (60-month sequence)