How to Protect my Assets from being Liquidated?

A Health Factor Monitoring and Portfolio Adjustment Assistant on AAVE*

Guoxuan Sun
Dept. of Engineering
CUHK
1155238658@link.cuhk.edu.hk

Chenwei Xu
Dept. of Engineering
CUHK
1155241558@link.cuhk.edu.hk

Siyu Huang
Dept. of Engineering
CUHK
1155241476@link.cuhk.edu.hk

Yexin Li
Dept. of Engineering
CUHK
1155180024@link.cuhk.edu.hk

Jialiang Zhou
Dept. of Engineering
CUHK
1155244677@link.cuhk.edu.hk

Abstract

The rapid expansion of Decentralized Finance (DeFi) has highlighted the critical challenge of liquidation risk in over-collateralized lending protocols like Aave. Existing risk management tools typically rely on static Health Factor (HF) thresholds, failing to account for systemic market fragility and high-frequency volatility. This study introduces an "Intelligent Risk Dual-Core Engine," a dynamic portfolio adjustment assistant designed to prevent liquidation while optimizing capital efficiency. We propose a multi-dimensional risk identification framework that synthesizes a macro Systemic Risk Indicator (R_I) with micro-level position metrics, utilizing Long Short-Term Memory (LSTM) networks for high-precision HF forecasting. Based on risk severity, the system deploys a tiered strategy: a Comparative Strategy for medium-risk stabilization and a Net Monetary Value (NMV) Optimization Strategy for high-risk crises. Furthermore, we implement a "Bridge Paradigm" architecture that couples off-chain algorithmic decision-making with on-chain atomic execution via a custom Solidity contract. Extensive simulations using historical Aave V3 data demonstrate that the proposed system effectively eliminates liquidation risk in volatile scenarios where traditional static monitoring fails, providing a robust solution for automated DeFi asset protection.

Keywords: Decentralized Finance (DeFi); Liquidation Risk Management; Aave Protocol; Cryptocurrency Price Prediction; Portfolio Optimization; Smart Contracts

*Project Source Code: `https://github.com/williamtage5/AAVE-Health-Factor-Monitoring-and-Portfolio-Adjustment-Assistant`

1 Mission Definition and Report Roadmap

1.1 Problem Context and Motivation

The decentralized finance (DeFi) ecosystem has experienced exponential growth, fundamentally reshaping global liquidity markets. At the forefront of this revolution are over-collateralized lending protocols like Aave, which allow users to monetize their assets without selling them. However, this financial innovation introduces a critical, asymmetric risk: **Liquidation**.

Unlike traditional finance, where margin calls often involve a grace period or human intervention, DeFi liquidations are atomic, irreversible, and executed by automated bots. When a user's collateral value drops relative to their debt—specifically, when their Health Factor (HF) falls below 1.0—smart contracts automatically auction off the collateral at a discount. This mechanism serves as the protocol's immune system to ensure solvency, but for the borrower, it represents a catastrophic financial loss, often exacerbated by the volatile nature of cryptocurrency markets.

The core problem this project addresses is the **"monitoring gap"** faced by individual investors. The crypto market operates 24/7 with high-frequency volatility. A borrower cannot physically monitor their position continuously. Existing solutions, such as static threshold alerts (e.g., "email me if $HF < 1.1$"), fail to capture the *velocity* of market crashes and often suffer from "alert fatigue." Furthermore, manual intervention is frequently thwarted by network congestion or high gas fees during market panics.

Therefore, there is an urgent need for an autonomous system that can not only "monitor" but also "predict" and "act"—bridging the gap between off-chain intelligence and on-chain execution to protect user assets proactively.

1.2 Project Mission and Objectives

The primary mission of this study is to design and prototype an **"Intelligent Risk Dual-Core Engine."** This system aims to transform risk management from a passive, reactive process into an active, predictive one.

Specifically, our objectives are threefold:

1. **Dynamic Risk Identification:** To move beyond static thresholds by developing a multi-factor risk model that synthesizes macro-systemic fragility (Systemic Risk Indicator, R_I) with micro-level position metrics (Health Factor Velocity).

2. **Predictive Intelligence:** To leverage Machine Learning (specifically LSTM networks) to forecast asset price volatility and Health Factor trends, enabling the system to anticipate insolvency before it occurs.

3. **Automated & Optimal Execution:** To implement a "Bridge Paradigm" architecture that connects off-chain decision-making with on-chain smart contracts, ensuring that portfolio adjustments (repayment or collateralization) are executed atomically and optimally (maximizing Net Monetary Value).

1.3 Report Roadmap and Deliverable Compliance

This report documents the end-to-end development of the proposed system. It is structured logically to address the specific requirements of the project assessment rubrics. The

content organization is as follows:

- **Section 2 (Background & Existing Solutions):** This section provides the necessary context regarding the Aave protocol. It critically analyzes *current practices* (manual monitoring, static bots) and their limitations. Furthermore, it discusses the *role of blockchain technology*, elaborating on how it mitigates trust issues while honestly addressing its *weaknesses*, such as oracle latency and gas fee volatility.
- **Section 3 (Strategy Design):** Here, we detail the core logic of our solution. We introduce the "Multi-Factor Dynamic Risk Identification Model" and the "Portfolio Adjustment Strategy." This section fulfills the requirement to describe the *functions of the developed prototype* and how they address the identified challenges through a tiered risk response mechanism.
- **Section 4 (Asset Price Prediction):** Recognizing that effective risk management requires foresight, this section details our technical implementation of price forecasting models. It compares XGBoost, LSTM, and Transformer architectures to justify our model selection for the prototype.
- **Section 5 (Data Engineering):** This section explains our methodology for acquiring high-fidelity data using *The Graph* and off-chain APIs. It establishes the rigorous data foundation required for reliable simulation.
- **Section 6 (Simulation & Validation):** We present empirical evidence of the system's efficacy. Through extensive backtesting on historical Aave V3 data, we demonstrate how the prototype would have performed in real-world crisis scenarios, providing the *detailed explanation of the logic of execution.*
- **Section 7 (On-chain Prototype & UI):** Finally, we showcase the tangible deliverables, including the "Bridge Paradigm" smart contract architecture and the user interface implementation, offering a glimpse into the *future outlook* of automated DeFi risk management.

By following this structure, the report aims to provide a comprehensive, rigorous, and technically sound solution to the problem of DeFi liquidation risk.

2 Background and Existing Solutions

2.1 Introduction to Aave

2.1.1 Overview

Aave is a decentralized, non-custodial liquidity protocol that enables users to supply digital assets to earn interest or to borrow assets by posting over-collateralized positions. Originally launched in 2017 as ETHLend under a peer-to-peer lending model, the platform rebranded to Aave in 2018 and transitioned to a liquidity-pool architecture. This structural shift significantly improved capital efficiency, risk management, and automation of lending operations. Aave has since expanded to multiple blockchains (e.g., Ethereum, Polygon, Avalanche) and introduced innovations such as flash loans, credit delegation, and collateral swaps, establishing its role as a foundational component of the decentralized finance ecosystem.

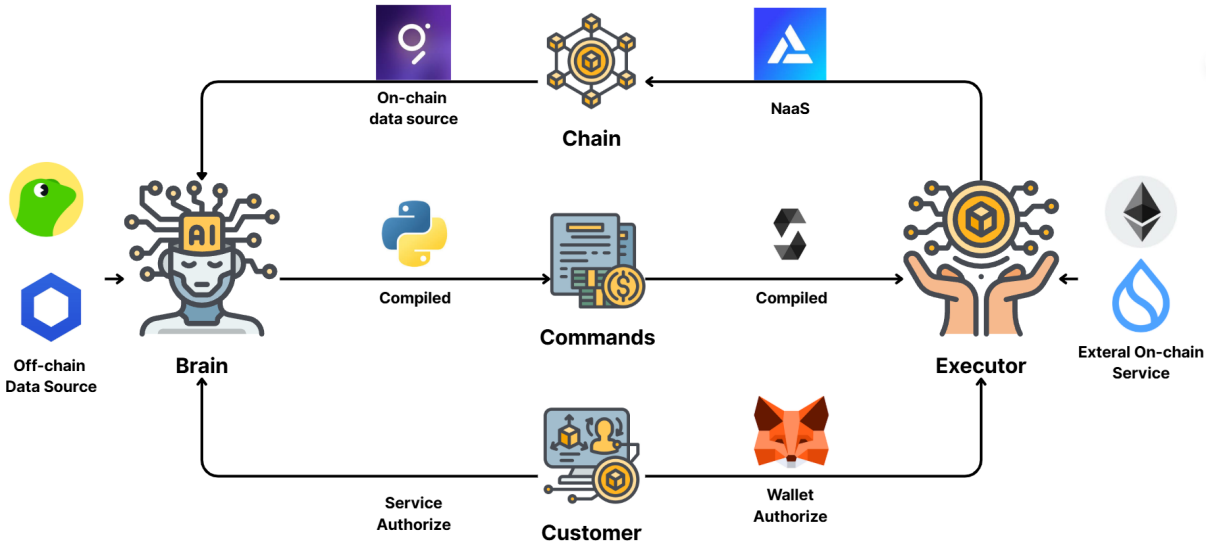


Figure 1: Project Construction

2.1.2 Supported Asset Types and Risk Parameters

Aave supports a wide spectrum of digital assets that vary in liquidity, volatility, and risk profiles. These assets are categorized as shown in Table1.

Table 1: Asset Categories in Aave

Category	Description	Examples
Protocol Tokens	Tokens used for governance.	AAVE, MKR, UNI, etc.
Major Crypto	Highly liquid, widely used crypto assets.	BTC, ETH, weETH, etc.
Stablecoins	Tokens designed to be value-stable..	USDT, DAI, GHO, etc.
Interest Tokens	Derivative or interest-accruing assets.	aUSDC, aETH, etc.
Long-Tail Assets	Higher-risk altcoins supported selectively.	LINK, ENS, etc.

Each asset is governed by a set of transparent, on-chain risk parameters including Loan-to-Value (LTV), Liquidation Threshold (LT), Liquidation Bonus, and its eligibility as collateral. These parameters determine borrowing capacity, liquidation conditions, and systemic risk exposure.

2.1.3 Lending and Borrowing Mechanism

Users supplying assets receive interest-bearing aTokens that automatically accumulate yield in real time. Borrowing is fully over-collateralized: users must deposit collateral whose adjusted value exceeds the borrowed amount. Two interest-rate modes—stable and variable—accommodate different risk and strategy preferences. Although Aave is not a decentralized exchange, users may conduct implicit asset swaps by borrowing one asset against another and later repaying the position.

A key systemic metric is the Health Factor (HF), defined as:

$$HF = \frac{\sum_i (V_{c,i} \cdot LT_i)}{B} \tag{1}$$

Positions remain safe when $HF > 1$; once $HF \leq 1$, the position becomes eligible for liquidation. The HF therefore aggregates collateral quality and borrowing exposure into a single risk indicator.

2.1.4 Liquidation Mechanism and Systemic Role

Aave employs a decentralized liquidation mechanism in which any external actor (liquidator) may repay part of an under-collateralized borrower's debt in exchange for collateral at a discounted rate, known as the liquidation bonus. This process is enabled when the borrower's Health Factor falls to 1.0 or below.

Liquidation simultaneously protects protocol solvency and introduces economic consequences for borrowers who maintain insufficient collateral buffers. For liquidators, the discounted collateral provides an arbitrage incentive, compensating for execution and market risks. The combined mechanism ensures that the lending pool remains fully collateralized even in volatile conditions.

2.2 Existing Practices and Challenges

To contextualize the necessity of an intelligent agent, it is essential to review how market participants currently manage liquidation risk in decentralized lending. The existing landscape is dominated by three primary approaches, each with significant limitations.

2.2.1 Current Risk Management Practices

- **Manual Monitoring:** The most rudimentary method involves users physically monitoring their Health Factor (HF) via the protocol's frontend interface. This relies entirely on human vigilance.
- **Static Alert Systems:** Many users employ third-party notification services (e.g., email or Telegram bots) that trigger an alert when the HF breaches a user-defined threshold (e.g., $HF < 1.1$). This approach is passive, requiring the user to receive the message and manually execute a transaction.
- **Basic Rule-Based Bots:** More sophisticated users may deploy simple scripts that automatically repay debt when a fixed numeric threshold is reached. These bots operate on rigid logic: "If $HF < X$, then Repay."

2.2.2 Critical Challenges

Despite these tools, liquidation remains a prevalent issue due to inherent structural flaws in current practices:

- **The Rigidity vs. Latency Trade-off:** Static thresholds are insensitive to market context. A threshold of 1.1 might be too conservative in a stable market (wasting capital efficiency) but too risky in a flash crash. Furthermore, manual intervention is impossible to sustain 24/7, creating a "monitoring gap" during sleep or work hours.
- **Alert Fatigue and Execution Risk:** Passive alerts often fail because they rely on the user's ability to react instantly. In times of extreme network congestion,

even if a user reacts immediately, their transaction may be pending for hours due to low gas fees, leading to unavoidable liquidation.

2.3 The Role of Blockchain Technology: Mitigation and Weaknesses

Our proposed solution leverages blockchain smart contracts to address these challenges. However, adopting this technology introduces a distinct set of trade-offs that must be critically evaluated.

2.3.1 Mitigation of Identified Challenges

Blockchain technology fundamentally alters the risk management paradigm through two key mechanisms:

- **Atomic Execution:** Unlike traditional banking systems where transfers and settlements are separate steps, smart contract operations are atomic. A risk management transaction (e.g., swapping collateral to repay debt) either executes completely or fails completely. This eliminates "partial execution risk," ensuring that the user is never left in an intermediate, vulnerable state.
- **Trustless Automation:** By deploying an on-chain verification contract (as detailed in the Prototype section), users can authorize a bot to execute specific protective actions without handing over custody of their private keys. This resolves the trust issue inherent in centralized asset management services.

2.3.2 Weaknesses and Downsides

While powerful, the reliance on blockchain infrastructure for real-time risk management is not without significant downsides. This study identifies two primary technological constraints:

- **Oracle Latency and Discreteness:** Blockchains are not continuous time systems; they are discrete state machines updated block-by-block (e.g., every 12 seconds on Ethereum). Furthermore, Chainlink oracles typically update prices only when a deviation threshold (e.g., 0.5%) is met. Consequently, in a "black swan" event where asset prices crash 10% in a single minute, the on-chain price feed may lag behind the real market price. This "blind spot" means a protective bot relying solely on on-chain data might react too late.
- **Gas Fee Volatility and Economic Viability:** Network congestion typically correlates with market volatility. Precisely when a user needs protection most (during a market crash), gas fees tend to spike exponentially. For smaller portfolios, the cost of the protective transaction (Gas Fee) might exceed the potential loss from the liquidation penalty. This economic irrationality creates a "protection floor," below which automated risk management becomes financially unviable on Layer 1 blockchains.

3 Portfolio Adjustment Strategy Design

3.1 Multi-Factor Dynamic Risk Identification Model

This section details the core strategy logic of our automated bot. It explains how we build a composite model that uses both macro systemic risk data and micro individual position data to decide exactly when to trigger preemptive deleveraging.

3.1.1 Motivation: Beyond Static Thresholds

Traditional risk management in DeFi lending often relies on monitoring a single Health Factor (HF). This approach usually sets a static, fixed safety buffer (e.g., watching if HF nears 1.1) as the only warning signal. However, our analysis shows this reliance on a single static indicator has significant limitations in complex markets:

- **Insensitivity to Macro Conditions:** A static number cannot distinguish between market states. A high HF position dropping slowly in a calm market is very different from one dropping sharply during a liquidity crisis. Strategies ignoring the macro dimension often react too late when systemic risks erupt.
- **Data Latency Risks:** On-chain HF calculations depend on oracle price feeds. In extreme market crashes, spot prices on centralized exchanges can fall significantly faster than on-chain oracle updates. A strategy relying only on lagged on-chain data may fail to act before irreversible liquidation happens.

Therefore, a robust strategy needs a multi-dimensional view that quantifies external systemic fragility and captures instant individual position changes.

3.1.2 Quantifying Systemic Risk: The R_I Indicator

To add a macro risk dimension, we draw upon the theoretical framework established in recent academic research on DeFi risk measurement, specifically the work of Bertomeu, Martin, and Sall (2024) in their paper "Measuring DeFi Risk". Their research argues that core protocol vulnerability arises specifically from asset-liability structural mismatches—namely, borrowing "rigid" liabilities (stablecoins whose value is fixed at \$1) against volatile collateral assets.

We adopted their proposed risk indicator, R_I, to act as a practical "thermometer" for AAVE protocol fragility.

According to Bertomeu et al., the R_I is defined as the ratio of total rigid debt to the **risk-weighted** net collateral value. Crucially, their derivation explicitly incorporates the protocol's risk parameters:

$$R_I = \frac{\text{Total Rigid Debt (Total Stablecoin Borrows)}}{\sum(\text{Volatile Collateral Market Value}_i \times \text{Liquidation Threshold}_i)} \tag{2}$$

By including the Liquidation Threshold (LT) in the denominator, this metric accurately reflects the *usable* collateral value available to back debts before liquidation triggers are hit, rather than just raw market value.

The core principle of this metric is to isolate the "scissor effect" risk: when market prices crash, the risk-adjusted collateral value (the denominator) shrinks rapidly, while

stablecoin debt values (the numerator) remain rigid. A high R_I indicates the protocol is highly sensitive to price shocks.

Threshold Setting: Instead of an arbitrary value, we ground our risk threshold in the empirical data provided by the literature. Referring to the summary statistics in the paper (Table 1), the historical **median** value of R_I is observed to be **0.63**. Consequently, we set $R_I > 0.63$ as our baseline warning threshold for systemic fragility, indicating that the market has entered a state more fragile than the historical average.

3.1.3 Multi-Factor Dynamic Measurement of Individual Risk

The macro R_I indicator provides context, but the final decision must consider the individual position status. We introduce two key micro-dynamic indicators to assess individual risk from long-term trend and instantaneous speed perspectives.

Long-term Position Trend: Weighted Predicted HF (`Weighted_HF`) To assess solvency trends over time and smooth out short-term noise, we use a weighted predicted HF over a future window. This reflects the average expected health level under our prediction model.

- **Risk Threshold Setting (`HF_Thresh`):** Instead of subjective fixed values, we use statistical methods for thresholds. We define the threshold based on the historical distribution of Health Factors (e.g., q30, the bottom 30th percentile). When `Weighted_HF` < `HF_Thresh` (q30), it indicates the long-term trend has entered a historically low-risk zone.

Instantaneous Position Speed: HF Percentage Change (`HF_pct_change`) To address oracle latency and capture instant market shocks, we introduce the dynamic `HF_pct_change` metric. It measures the speed and direction of HF change per unit time.

The calculation formula is:

$$HF\_pct\_change_t = \frac{HF_t - HF_{t-1}}{HF_{t-1}} \tag{3}$$

Positive values mean improvement; negative values mean deterioration.

- **Speed Threshold Setting (`HF_Vel_Thresh`):** The core function of this metric is identifying "rapid crashes." We set a statistical negative speed threshold based on historical data (e.g., the q20 percentile, such as a drop exceeding 2% in a short time). When `HF_pct_change` < `HF_Vel_Thresh` (q20), it signals a violent instantaneous deterioration, regardless of the current absolute HF level.

3.1.4 Comprehensive Decision Model: Dynamic Risk Matrix

Our final strategy integrates these three dimensions—macro systemic fragility (R_I), micro long-term trend (`Weighted_HF`), and micro instant speed (`HF_pct_change`)—into a single decision matrix. As detailed in Table 2, this matrix classifies market states into three levels with corresponding actions:

1. **High Risk**

   This state represents extreme danger, triggering immediate bot deleveraging. It is triggered if either of the following conditions is met:

- **Condition A (Liquidated State):** `Weighted_HF` < 1.0. The position has effectively broken the liquidation line.
- **Condition B (Critical Combination):** System Fragile ($R_I > 0.63$) **AND** Position Crashing Fast (`HF_pct_change` < `HF_Vel_Thresh`). This specific combination captures the most dangerous scenario: a sharp individual deterioration occurring during extreme macro fragility, indicating a very high probability of irreversible liquidation.

2. **Medium Risk**

   This indicates risk factors are building up. The bot enters high alert and prepares resources but does not execute immediately. It is triggered if any single risk indicator is active (without meeting High Risk conditions):

   - **Condition A (System Warning):** $R_I > 0.63$ (Macro environment fragile).
   - **Condition B (Speed Warning):** `HF_pct_change` < `HF_Vel_Thresh` (Instant individual crash).
   - **Condition C (Trend Warning):** `Weighted_HF` < `HF_Thresh` (q30) (Long-term trend low).

3. **Low Risk**

   When neither High nor Medium risk conditions are met, the system is deemed safe. The macro environment is stable, and individual trend and speed are within normal ranges. The bot maintains passive monitoring status.

Table 2: Dynamic Risk Matrix: Triggering Logic and Rationale

Risk Level	**Triggering Logic**	**Rationale**
Low Risk	**ALL of the following must be true:** 1. System Stability: $R_I \leq 0.63$ 2. Position Velocity: `HF_pct_change` $\geq$ `HF_Vel_Thresh` 3. Position Trend: `Weighted_HF` $\geq$ `HF_Thresh` (P30)	System, Velocity, and Level are all within safe parameters.
Medium Risk	**(Not High Risk) AND ANY of the following is true:** 1. System Warning: $R_I > 0.63$ 2. Speed Warning: `HF_pct_change` < `HF_Vel_Thresh` 3. Trend Warning: `Weighted_HF` < `HF_Thresh` (P30)	At least one indicator shows potential risk, signaling a need for preparation.
High Risk	**ANY of the following is true:** 1. Liquidated State: `Weighted_HF` < 1.0 2. Critical Combination: ($R_I > 0.63$) **AND** (`HF_pct_change` < `HF_Vel_Thresh`)	User is crashing fast while the market is fragile, or position is already liquidated. Immediate action required.

By using this multi-factor dynamic model, our strategy addresses the blind spots of static thresholds. It effectively uses macro and micro data to ensure decisive action when necessary while avoiding overreaction during non-critical times.

3.2 Portfolio Adjustment Operation Model

3.2.1 Core Execution Logic of the Strategy

The core principle of the execution layer is making decisions based on real-time data and executing operations according to protocol rules, following the process of **"Preprocessing - Core Operations - Result Verification"** in sequence.

3.2.2 Priority Order of Collateral Repayment

Modern DeFi protocols (such as Aave) permit multiple assets as collateral and enable borrowing of various assets, presenting multidimensional and interrelated risks. In complex multi-asset collateral scenarios, liquidation tools must determine which debts to repay and which collateral to liquidate. This strategy prioritizes repaying assets that "most significantly boost the health factor while keeping operational costs the lowest and risks minimal". The core ranking principles are as follows (in descending order).

Highest priority: Largest unit improvement to health factor Debt repayment and collateral replenishment exert opposite effects on the health factor, but both contribute to its enhancement. To formalize the portfolio rebalancing decision-making process, we define the following decision variables for the two operations:

- X_{r} is the amount allocated to repay high-priority debt.
- X_{s} is the value allocated to supply core collateral.

Debt repayment improves the health factor by reducing the denominator in its calculation formula, while collateral replenishment boosts it by increasing the numerator:

$$HF_{\mathrm{after}}(X_{\mathrm{r}}, X_{\mathrm{s}}) = \frac{(Collateral_{before} + X_{\mathrm{s}}) \times LT}{Debt_{before} - X_{\mathrm{r}}} \tag{4}$$

where:

- $Collateral_{before}$ refers to the total value of the borrower's collateral before operation.
- $Debt_{before}$ refers to the total debt of the borrower before operation.
- LT refers to the liquidation threshold set for this type of asset (a percentage).

Given their distinct mechanisms for improving the HF, it is necessary to design a dedicated efficiency indicator to compare these two adjustment methods. Specifically, this indicator measures "how much the health factor increases for every 1 ETH of debt repaid."

$$HF\_unit\_improvement = \frac{HF_{\mathrm{after}} - HF_{\mathrm{before}}}{ETH} \times 100\% \tag{5}$$

where:

- *HF_unit_improvement* refers to operation's unit improvement to health factor.
- ETH serves as the unit of account for debt.

Operations corresponding to assets that deliver the greatest HF improvement per unit of debt repaid are prioritized in execution.

Prioritize Assets with Lowest Risk Levels We prioritize handling high-risk assets to prevent subsequent risk escalation. We first repay "variable-rate debt"(subject to significant interest rate fluctuations), and then settle "fixed-rate debt". We prioritize repaying "isolated asset debt"(for which debt ceilings may be adjusted) before settling "core asset debt".

User Asset Allocation Preferences In practice, a health factor below 1 does not automatically trigger liquidation. This may stem from liquidators' decisions, as we respect user preferences. If a user marks "Hold ETH long-term, do not liquidate", other collateral is liquidated first. If a user sets "Prioritize using idle USDC for repayment", USDC is used first even if other assets have lower transaction costs.

3.3 Strategy Execution Design

For R1(risk indicator) at different risk levels, targeted portfolio rebalancing strategies are triggered to adjust the health factor to a stable level. When selecting between the "Debt Repayment" or "Collateral Supplementation" scenarios, operational feasibility is determined based on the user's assets or debts.

Strategy implementation is divided into two distinct trigger points: The first is activated when R1 transitions from a low-risk to a medium-risk state; the second is activated when R1 shifts from low-risk to high-risk, or from medium-risk to high-risk.

3.3.1 Strategy One

When R1 transitions from a low-risk to a medium-risk state, the more effective method between two options, "Debt Repayment" or "Collateral Supplementation", is adopted. The process of "partial debt repayment" is as follows: Step one: Confirm the user's idle assets and verify that their wallet holds sufficient coins for repayment to prevent failed transfers. Step two: Determine the repayment amount by calculating "repaying 5 % -10% of debt to restore the health factor". Step 3: Execute the repayment interaction using Aave's repay function, settling the debt with the user's idle stablecoins.

The process of "collateral supplementation" is as follows: Step 1: Confirm that the user holds eligible assets for collateral supplementation. Step 2: Execute the collateral top-up by invoking Aave's supply function to deposit the user's idle assets as collateral. Key parameters include the collateral address and the top-up amount.

3.3.2 Strategy Two

When R1 transitions from a low-risk to a high-risk state, or from a medium-risk to a high-risk state, the following actions shall be executed: fully repay high-priority debt and supplement core collateral.

The core objective at this stage is to raise the HF to a safe threshold (i.e., a level sufficiently above the liquidation line) through the combined strategy of "full repayment of high-priority debt and supplementation of core collateral", while avoiding liquidation. The fundamental logic is to use an optimization model to find the perfect split between fully repaying high-priority debt and supplementing core collateral.

Operational optimization involves constructing an objective function to measure this strategy's effectiveness in improving the Health Factor. This approach monetizes risk reduction benefits, formulating the objective function in terms of Net Monetary Value (NMV). The objective function is defined as:

$$\max Z = \lambda_{HF} \cdot (HF_{after} - HF_{before}) - Cost_{total} \tag{6}$$

where:

- Z yields the net benefit in USD.
- λ_{HF} represents HF clearing trigger threshold (i.e., HF=1.0).
- $Cost_{total}$ is the cost incurred during asset transactions. Its calculation formula is given before.

By Solving the function, we will find the optimal ratio between supplementing collateral and repaying debts when Z reaches its maximum value.

3.4 Evaluation Metrics Design

When executing portfolio adjustment operations, it is necessary to establish metrics to measure the impact of such adjustments on future fluctuations of the health factor, thereby supporting strategy selection. This study establishes three metrics to measure the merits of different operations.

3.4.1 Magnitude of Health Factor Improvement: A Core Risk Metric

Data sources include Aave's collateral and debt balances. By integrating liquidation thresholds and asset prices, the health factor (HF) values before and after the operation are calculated. This enables us to further compute the improvement in HF.

$$HF\_increase\_rate = \frac{(HF_{after} - HF_{before})}{HF_{before}} \times 100\% \tag{7}$$

If the calculated improvement magnitude is greater than zero, the proposed strategy achieves effective risk reduction, with higher positive values denoting better performance. If the value is less than or equal to zero, the operation fails to achieve the intended risk reduction effect.

3.4.2 Reduction in Liquidation Probability: A Risk Warning Indicator

This serves as the core innovative indicator of our strategy, which quantifies the probability that a user's HF will drop below the liquidation threshold (i.e., HF=1.0) within a future T-day horizon. For model selection, we follow the approach proposed by Belenko et al. (2025), which assumes that collateral prices follow a Geometric Brownian

Motion (GBM). We abandon traditional financial methods such as NORM.DIST and the reflection principle—as Belenko et al. (2025) demonstrated that these methods exhibit errors in medium-to-long-term risk assessment—in favor of the analytical solution to the Inverse Gaussian Distribution. This method enables the precise calculation of the liquidation probability within the T-day horizon.

The probability P of liquidation occurring before the time range T (6 hours) is calculated using the cumulative distribution function (CDF) of the Inverse Gaussian Distribution. The function is defined as:

$$\mathbb{P}(liquidation \leq T) = \Phi\left(\frac{B + \mu_{\text{eff}} T}{\sigma\sqrt{T}}\right) + e^{-\frac{2B\mu_{\text{eff}}}{\sigma^2}} \Phi\left(\frac{-B + \mu_{\text{eff}} T}{\sigma\sqrt{T}}\right) \tag{8}$$

where:

- S represents initial health factor.
- $S_{liquidation}$ represents HF clearing trigger threshold (i.e., HF=1.0).
- σ is HF hourly volatility (obtained from Aave price history or third-party oracle).
- $\mu_{\text{eff}} = -\frac{\sigma^2}{2}$ is the effective negative drift term generated by GBM logarithmic transformation.
- $B = \ln\left(\frac{S_0}{S_{\text{liquidation}}}\right)$ is the logarithmic price distance.

All statistical measures are simulated based on the fluctuation statistics of HF itself. The formula for calculating the decrease in the probability of liquidation is defined as:

$$\mathbb{P}_{decrease} = \frac{\mathbb{P}(liquidation \leq T)_{before} - \mathbb{P}(liquidation \leq T)_{after}}{\mathbb{P}(liquidation \leq T)_{before}} \times 100\% \tag{9}$$

The greater the decline in liquidation rate, the more effective the adjustment strategy becomes.

3.4.3 Operational Costs

Additionally, portfolio adjustment operations must take into account the operational cost. Given the complexity of implementation and data cleansing, this study narrows the research scope and simplifies the measurement of associated costs. Gas Fees and other associated implicit costs are excluded. Instead, only price slippage costs are considered as the core operational cost of portfolio adjustment. Price slippage cost (i.e., slippage loss) refers to the economic loss incurred by the deviation between the oracle price and the liquidation execution price.

$$C_{slippage} = (P_{predicted} - P_{actual}) \times Q \tag{10}$$

where:

- $C_{slippage}$ refers to the price slippage cost.
- $P_{predicted}$ refers to the oracle price we predicted.
- P_{actual} refers to the actual liquidation execution price.

- Q refers to transaction quantities.

If the price slippage cost is positive, it indicates that the portfolio adjustment operation achieves positive net effectiveness.

4 Assets Price Prediction

Price prediction is our core section, where we aim to predict fluctuations in health factors by forecasting cryptocurrency prices, thereby incorporating them into our overall strategy.

4.1 Data Engineering and Factors Selection

The models are developed using hourly cryptocurrency price data as the primary input. To fully capture market dynamics, an extensive feature engineering process was conducted, resulting in a comprehensive input feature set. To improve the precision of the prediction, we incorporate 11 technical indicators(Table3) as input features.

Table 3: Factors Selection

Indicators	Description
Return	Daily percentage change of closing price
MA5	5-day moving average of closing price
MA10	10-day moving average of closing price
MA20	20-day moving average of closing price
RSI14	14-day Relative Strength Index measuring momentum
MACD	Difference between 12-day and 26-day EMAs
Signal	9-day EMA of the MACD
MACD_Hist	Difference between MACD and Signal
BB_Mid	20-day moving average (Bollinger middle band)
BB_Upper	Upper Bollinger band (middle $+ 2\sigma$)
BB_Lower	Lower Bollinger band (middle $- 2\sigma$)

4.2 Model Construction and Training

This study develops three forecasting models—XGBoost, LSTM, and a Transformer-based neural architecture—to predict hourly cryptocurrency closing prices.

- XGBoost

  The first model, XGBoost, reshapes each sliding window into a fixed-length feature vector and gradient-boosted decision trees effectively capture nonlinear interactions among lagged variables. In this study, the XGBoost regressor is trained with 500 boosting rounds, a learning rate of 0.02, and a maximum tree depth of 6. L1 and L2 regularization terms of 0.2 and 0.4, respectively, are imposed to reduce overfitting. To enhance model robustness, five independent models with different random seeds are trained using the same configuration, and their predictions are averaged to form an ensemble that mitigates variance inherent in individual tree-based learners.

- LSTM

  The second model is a two-layer Long Short-Term Memory (LSTM) network implemented in PyTorch. Unlike XGBoost, the LSTM operates directly on sequential multivariate inputs and employs memory cells to retain long-range temporal dependencies. The network consists of two stacked LSTM layers, each with a hidden size of 64 units and a dropout rate of 0.2 applied between layers to prevent overfitting. Input sequences, containing 24 past hours of technical indicators, are processed sequentially, and the hidden state from the final time step is fed into a fully connected output layer to generate the predicted closing price. The model is optimized using the Adam optimizer with an initial learning rate of 1×10^{-3}, and mini-batch training over 60 epochs is conducted to ensure stable convergence, accompanied by a learning-rate scheduler that gradually decays the step size.

- Transformer

  The third model in this study is a Transformer-based encoder architecture tailored for multivariate time-series forecasting. The network first projects the input technical indicators into a 128-dimensional latent space through a linear input layer. This representation is processed by a stack of four Transformer encoder layers, each consisting of eight-head self-attention and a position-wise feed-forward network with a hidden dimension of 256. A dropout rate of 0.1 is applied within both the attention and feed-forward sublayers, and layer normalization is incorporated to enhance numerical stability. After the encoder stack, the latent vector corresponding to the final time step is extracted and passed through a two-layer fully connected decoder, with GELU activation between layers, to produce the one-step-ahead forecast. The model is trained using the AdamW optimizer with an initial learning rate of 2×10^{-4} and weight decay of 1×10^{-4}. A cosine-annealing learning-rate scheduler with $T\_max = 50$ is applied to progressively reduce the learning rate during training. To prevent overfitting, early stopping with a patience of 15 epochs is employed, and the checkpoint with the lowest training loss is retained for evaluation.

All models are trained chronologically to avoid information leakage. The dataset is divided into training and testing segments according to time order, and a sliding window approach is used to construct input–output pairs. For neural models, the training process is conducted with shuffled mini-batches, while the overall temporal structure of the data is preserved across epochs. The loss function for both LSTM and Transformer is mean squared error (MSE), and model checkpoints are saved based on validation performance.

4.3 Model Evaluation and Comparison

To identify the model with the best predictive performance, we trained and backtested each of the three models three times using five cryptocurrencies, including cbBTC, cbETH, AAVE, EURC and weETH and the time period is from August 1, 2025, to October 30, 2025. We split the training set and test set in a 7:3 ratio and the backtesting is conducted by generating one-step-ahead predictions across the entire test period. The model performance is evaluated by calculating the mean and standard deviation of MSE and R^2 through multiple repeated experiments. The experiment results are shown in the table4.

Table 4: Model Performance

Cryptocurrency	Model	MSE	R^2
cbBTC	XGBoost	$1369197.1300 \pm 10174.6368$	0.9565 ± 0.0003
	LSTM	$303847.7767 \pm 19882.6597$	0.9903 ± 0.0006
	Transformer	$381676.2896 \pm 8927.9371$	0.9880 ± 0.0003
cbETH	XGBoost	2808.1467 ± 54.8541	0.9663 ± 0.0007
	LSTM	1420.2479 ± 183.2443	0.9829 ± 0.0021
	Transformer	1460.3586 ± 78.4767	0.9825 ± 0.0008
AAVE	XGBoost	597.4928 ± 1.0640	0.1028 ± 0.0015
	LSTM	11.9319 ± 3.0635	0.9821 ± 0.0043
	Transformer	62.2442 ± 22.9242	0.9066 ± 0.0345
EURC	XGBoost	$0.00000959 \pm 0.00000014$	0.8026 ± 0.0026
	LSTM	$0.00000122 \pm 0.00000006$	0.9748 ± 0.0013
	Transformer	$0.00000231 \pm 0.00000056$	0.9524 ± 0.0113
weETH	XGBoost	2749.5612 ± 51.5437	0.9647 ± 0.0006
	LSTM	1363.2635 ± 88.3432	0.9825 ± 0.0012
	Transformer	1525.1529 ± 137.3843	0.9804 ± 0.0016

The experimental results show that

1. XGBoost exhibits exceptional stability across all cryptocurrencies but yields the highest mean MSE.
2. LSTM and Transformer models show relatively larger MSE fluctuations but achieve lower mean MSE values.
3. For less volatile currencies like stablecoins, all models demonstrate stable performance.
4. For highly volatile assets like AAVE, sequence models show greater variability.

Across all experiments, the LSTM model demonstrates superior capability in capturing both short-term fluctuations and long-range dependencies, exhibiting smoother predictions and lower overall error. The overall performance of the transformer is comparable to that of LSTM, but its fitting quality is significantly lower than LSTM's for assets with higher volatility. The XGBoost ensemble, while competitive in stable market periods, is comparatively limited in modeling sequential structures. Ultimately, through comparative evaluation of model performance, the LSTM model was chosen for follow-up health factors prediction tasks.

5 Data Request

5.1 Related Tech

5.1.1 Decentralized Indexing and State Retrieval Architecture

Structural Limitations of Sequential Blockchain Querying Fundamentally, blockchain architectures, including Ethereum, are optimized for write availability, consensus

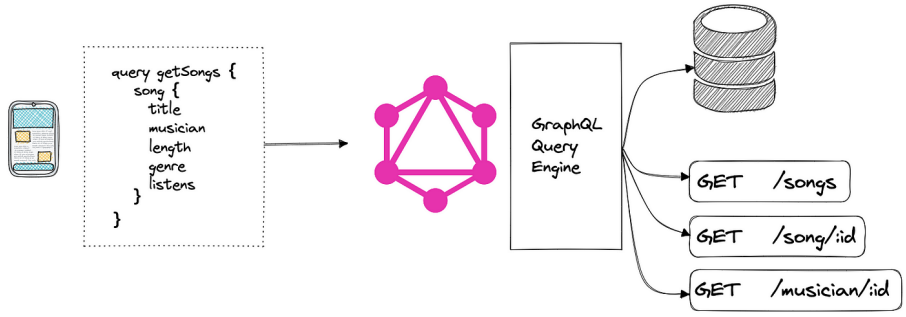


Figure 2: GraphQL Schematic Diagram

security, and cryptographic immutability, rather than for read efficiency. The underlying data structure resembles a singly linked list of blocks , where state transitions are recorded sequentially and hashed cryptographically. While this structure ensures the integrity of the ledger, it presents significant computational challenges for longitudinal data analysis—the study of state evolution over time. The standard interface for interacting with blockchain nodes, the Remote Procedure Call (RPC), is designed primarily for point-in-time state retrieval or transaction broadcasting, making it inherently unsuitable for historical analysis.

The primary limitation lies in the sequential nature of RPC querying. To reconstruct the historical trajectory of a specific variable, such as a user's collateral balance, over a defined period, an analyst must iteratively query the node at every individual block height within the target interval. This results in a linear time complexity of $O(N)$, where N is the number of blocks. For high-frequency analysis spanning extended periods, the sheer volume of HTTP requests required renders this approach computationally prohibitive and prone to latency and rate-limiting by node providers. Furthermore, RPC interfaces function similarly to key-value stores and lack the semantic richness of Relational Database Management Systems (RDBMS). They do not support complex filtering, aggregation, or the joining of distinct entities. For instance, a complex query such as "identify the top 100 users sorted by total debt at the moment of liquidation" cannot be executed directly via RPC; instead, it would require scanning the entire event log history, locally parsing Bloom filters, and decoding raw hexadecimal logs. This opacity barrier" necessitates the adoption of a dedicated indexing layer capable of transforming raw, unstructured chain data into organized, queryable schemas.

The Event-Driven Indexing Paradigm To address the architectural constraints of direct blockchain interaction, this study employs The Graph, a decentralized indexing protocol designed to organize blockchain data and make it easily accessible via GraphQL. The protocol functions as a middleware layer that abstracts the complexity of data extraction, allowing for the creation of open APIs, termed "subgraphs," which define specific data ingestion pipelines. By converting the unstructured, sequential ledger of the blockchain into a structured, relational database format, The Graph provides the necessary infrastructure to reconstruct complex financial histories, such as user liquidation arcs, with high precision and efficiency.

The operational workflow of a subgraph follows a rigorous event-driven paradigm. The process begins with a manifest file (YAML), which specifies the target smart contracts, such as the Aave Lending Pool, and the specific Ethereum Virtual Machine (EVM) events to monitor. Graph Nodes then scan the blockchain network for relevant transactions and

logs that match these criteria, ensuring that no relevant state change is omitted. Upon detecting a target event, the node triggers a mapping script compiled in WebAssembly (WASM), which contains the business logic required to transform raw event data into meaningful data types. For example, a raw `Deposit` event is parsed and converted into an updated user balance entity. This processed data is deterministically stored in an underlying PostgreSQL database, organized according to a pre-defined schema. Finally, the indexed data is exposed through a GraphQL endpoint, enabling clients to perform complex queries—including filtering, sorting, and pagination—with millisecond-level latency.

Deterministic State Retrieval via Block-Height Time Travel A critical capability required for retrospective financial analysis is the ability to query the state of the ledger at arbitrary points in history, a feature often termed "time-travel queries." While standard APIs typically return only the current head state of the blockchain, The Graph protocol leverages the immutable nature of the distributed ledger to enable deterministic state retrieval. By appending a specific block constraint—denoted as `block: { number: N }`—to a GraphQL query, the indexing node is instructed to disregard the latest state and instead retrieve data as it existed at the precise moment block N was validated.

Technically, this mechanism relies on the underlying infrastructure of "Archive Nodes," which retain the full history of the state trie for every block since the genesis. When a time-travel query is executed, the query engine traverses the versioned database to locate the entity state valid at the target block height. This ensures that the retrieved data—such as a user's collateral balance or the accumulated debt interest—is an exact, deterministic snapshot of the past, unaffected by subsequent state transitions. For the purpose of this study, this feature is indispensable. It allows for the reconstruction of the "pre-liquidation" environment, enabling the analysis to isolate the exact financial conditions that precipitated a liquidation event, free from the noise of the liquidation transaction itself.

5.1.2 Hybrid Data Acquisition and Synchronization Strategy

Precision of On-Chain Static Asset Snapshots The reconstruction of a user's financial health necessitates a strict dichotomy between the retrieval of asset quantities and asset valuations. For the former—specifically, the precise amounts of collateral deposited and debt incurred—this study relies exclusively on the on-chain data indexed by the Aave Subgraph. These values represent the "ground truth" of the ledger; they are the direct result of deterministic smart contract interactions and are not subject to external interpretation.

Utilizing the block-height time travel mechanism described previously, we capture the static state of a user's portfolio at the exact block preceding the liquidation event. This approach ensures that the "liability structure" of the portfolio—the exact number of tokens held and the accrued interest on debts up to that specific second—is mathematically identical to the state processed by the Aave protocol's validation logic. Reliance on any off-chain estimation for these quantities would introduce unacceptable measurement errors, as it would fail to account for the continuous, second-by-second compounding of variable interest rates intrinsic to DeFi lending protocols.

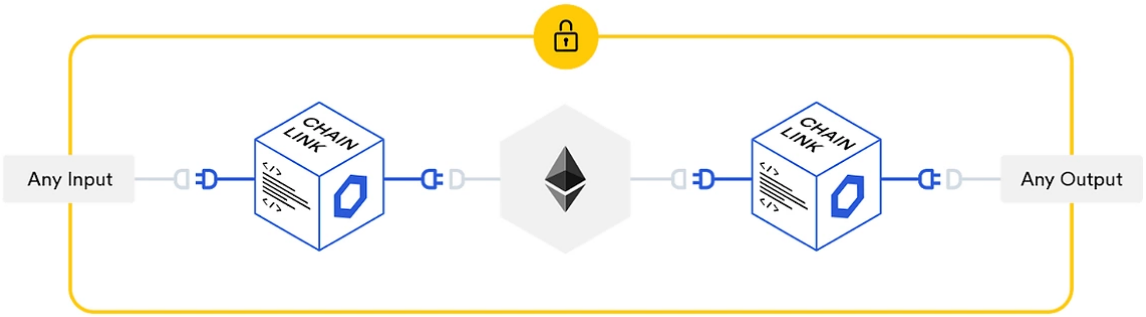


Figure 3: How Top Blockchain Oracles Enhance Their Data Offering

Mitigating Oracle Discontinuity via High-Frequency Off-Chain Data While asset quantities are retrieved from the blockchain to ensure precision, relying solely on on-chain data for asset *pricing* presents significant methodological challenges due to the mechanical limitations of blockchain oracles. Protocols like Aave typically employ Chainlink oracles, which update prices based on a "deviation threshold" (e.g., a 0.5% price change) or a "heartbeat" (e.g., every 24 hours). Consequently, on-chain price data exists as a discrete, step-function time series rather than a continuous curve

This sparsity can mask high-frequency volatility; a price might spike momentarily, trigger a liquidation, and revert before a new oracle update is committed to the block, or conversely, the oracle update might lag slightly behind market movements due to gas congestion.

To address this discontinuity and reconstruct a continuous timeline of risk, this study incorporates high-frequency off-chain market data from CoinGecko. By utilizing centralized exchange aggregated data, we generate a granular, continuous price vector that fills the gaps between sporadic on-chain oracle updates. This hybrid approach—combining the indisputable structural accuracy of on-chain balances with the high-resolution temporal continuity of off-chain pricing—allows for a more rigorous simulation of the health factor's trajectory, enabling the detection of momentary valuation dips that theoretically necessitated the liquidation but were not explicitly captured in the sparse historical logs of the oracle smart contracts.

Numeraire Standardization and Temporal Alignment The integration of disparate data sources—static blockchain snapshots and dynamic external price feeds—requires a rigorous synchronization framework to ensure temporal consistency. Since the user's asset quantities (Q) are treated as constant invariants over the defined event study window $[T_{start}, T_{end}]$, the synchronization process focuses on aligning the timestamped price vectors (P_t) retrieved from CoinGecko with the specific hourly intervals of the study period. This creates a coherent time-series matrix where every row represents a distinct state vector of the user's portfolio value.

Furthermore, to isolate idiosyncratic asset risk from systematic market volatility, this study applies a numeraire standardization process. While external APIs typically provide valuations in fiat currency (USD), DeFi protocols like Aave have historically relied on Ethereum (ETH) as a primary unit of account for internal risk calculations. Analyzing health factors based purely on USD prices can introduce "fiat noise"; for instance, if the entire market crashes against the dollar, the relative value between collateral (e.g.,

WBTC) and debt (e.g., ETH) might remain stable, leaving the health factor unaffected. To accurately model this protocol-native logic, we normalize all asset prices to an ETH standard. Mathematically, the price of an arbitrary asset i at time t is transformed according to Equation 11:

$$P_{i,t}^{ETH} = \frac{P_{i,t}^{USD}}{P_{ETH,t}^{USD}} \tag{11}$$

This normalization filters out broad market beta, ensuring that the simulated health factor fluctuations reflect genuine changes in the collateral-to-debt ratio rather than exogenous movements in the fiat exchange rate.

5.2 Deployment and Results

5.2.1 Infrastructure and Middleware Setup

To ensure the stability, security, and reproducibility of the data acquisition process, a dedicated local middleware layer was engineered utilizing the FastAPI framework. This middleware functions as a secure proxy gateway, effectively abstracting the complexities of authentication and connection management associated with The Graph's decentralized network. By encapsulating the Subgraph Studio API credentials within this server-side environment, the architecture prevents the exposure of sensitive keys in the client-side analysis scripts and manages request rate-limiting to adhere to network protocols.

The downstream data analysis pipeline is constructed in Python, leveraging the web3 library in Python for cryptographic interactions and the gql library for constructing strongly typed GraphQL queries. This pipeline interfaces exclusively with the local FastAPI endpoint, establishing a robust communication bridge. This decoupled architecture ensures that the analysis logic remains agnostic to the underlying transport variations of the decentralized indexing protocols.

5.2.2 Target Sampling and Temporal Boundary Definition

The construction of the dataset commences with the identification of a representative cohort of distressed entities. We executed a reverse-chronological query against the `liquidationCalls` entity of the Aave Subgraph to retrieve the 100 most recent liquidation events. For each unique entity u, the timestamp of this liquidation transaction defines the terminal boundary of the event study window, denoted as T_{end}.

To rigorously define the causal observation period, it is essential to identify the precise moment when the user's portfolio configuration last underwent an active state transition prior to the liquidation. We implemented a backtracking algorithm that executes a composite query across three distinct historical entities:

1. `reserves`: Captures standard capital interactions such as deposits, borrowings, and repayments.

2. `userEmodeSetHistory`: Tracks changes to Efficiency Mode (E-Mode) settings, which significantly alter liquidation thresholds.

3. `liquidationCallHistory`: Identifies any prior liquidation events that would reset the account state.

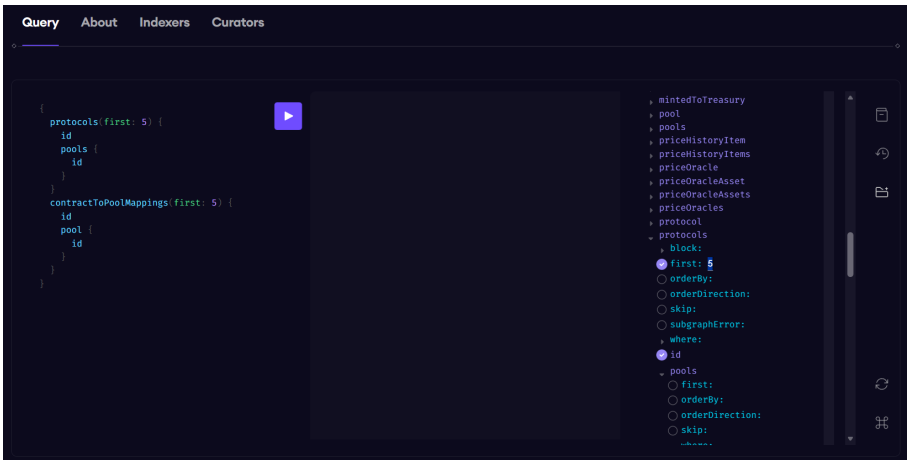
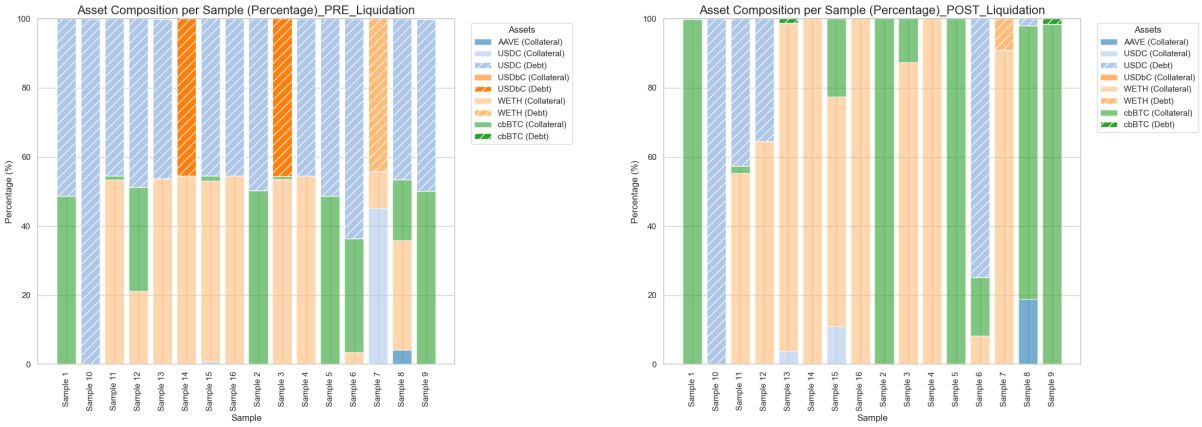


Figure 4: AAVE Request Playground Shema



(a) Before liquidation

(b) After Liquidation

Figure 5: Some Samples Portfolio

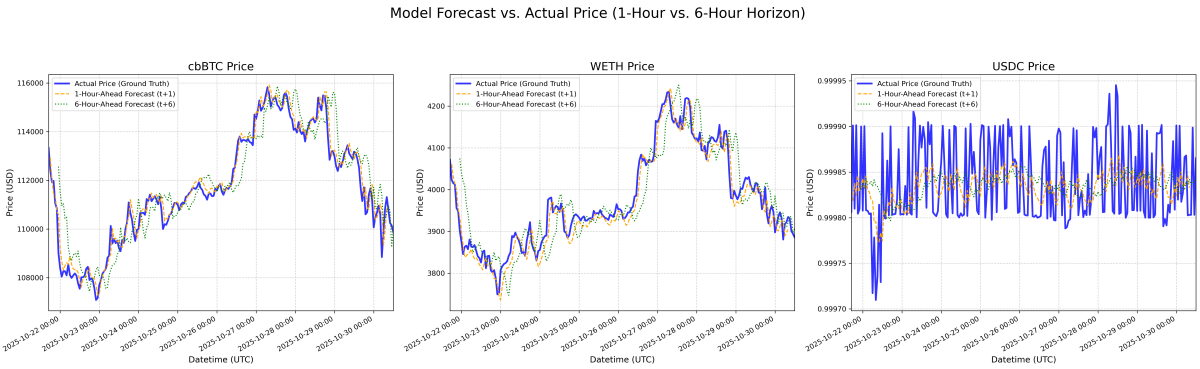


Figure 6: Portfolio Forecast in Different Horizon comparison for Target Address

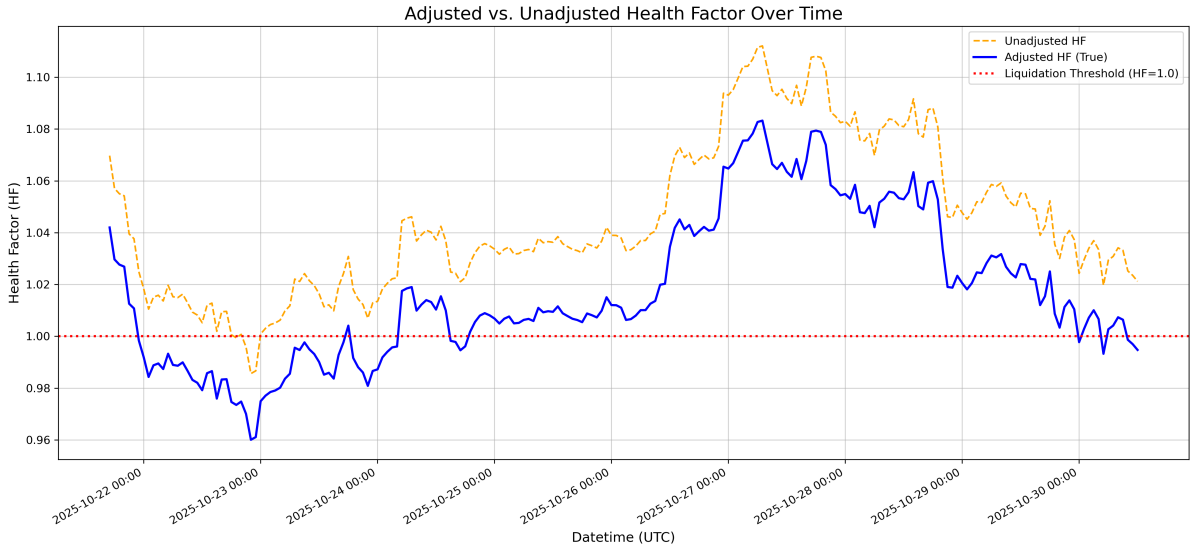


Figure 7: Adjusted and Unadjusted HF Prediction Comparasion for Target Address

The study window start time, T_{start}, is mathematically derived as the maximum timestamp among these potential antecedent events, conditional on being strictly less than T_{end}. Entities for which no historical lineage could be retrieved within the indexing depth—resulting in an undefined T_{start}—were rigorously excluded from the sample. This filtering process ensures that every observation in the final dataset represents a closed, fully deterministic time interval $[T_{start}, T_{end}]$ suitable for longitudinal reconstruction.

5.2.3 Pre-Liquidation State Reconstruction

Precise reconstruction of the financial conditions precipitating a default requires mapping the temporal timestamps of liquidation events to their corresponding immutable blockchain coordinates. We utilized the Etherscan API to resolve the transaction hash ($TxHash$) of each identified liquidation event into its specific block height, denoted as B_{liq}.

A critical methodological challenge in post-mortem analysis is avoiding "look-ahead bias." Querying the ledger state at B_{liq} would return data reflecting the post-execution state—where debt has already been repaid and collateral seized—thereby obscuring the specific leverage conditions that triggered the protocol's liquidation logic. To mitigate this, we employed the "N-1 Block Snapshot" technique. By executing time-travel queries

specifically targeting block height $B_{liq} - 1$, we successfully captured the exact liability structure (collateral composition and outstanding debt obligations) of the user at the precise moment immediately preceding the execution of the liquidation transaction. This approach ensures that the extracted state vector represents the genuine risk exposure faced by the user at the critical event boundary.

6 Strategies Simulation

6.1 Parameters Setup

6.1.1 Mathematical Modeling of Adjusted Liquidation Risk

Formal Definition of Protocol Risk Metrics The central metric governing the solvency of a position within the Aave protocol is the Health Factor (HF). It functions as a safety buffer, quantified as the ratio of the risk-adjusted value of collateral to the total value of outstanding debt. Formally, for a user u at time t, the Health Factor is defined as:

$$HF_{u,t} = \frac{\sum_{i \in \mathcal{C}} (Q_{i,u} \cdot P_{i,t} \cdot LT_i)}{\sum_{j \in \mathcal{D}} (Q_{j,u} \cdot P_{j,t})} \tag{12}$$

Where:

- $\mathcal{C}$ and $\mathcal{D}$ represent the sets of collateral assets and debt assets, respectively.
- $Q_{k,u}$ denotes the quantity (balance) of asset k held or borrowed by user u.
- $P_{k,t}$ denotes the market price of asset k at time t (normalized to ETH as per Section 3.2.3).
- LT_i is the *Liquidation Threshold* of the collateral asset i, a protocol-defined parameter representing the maximum loan-to-value ratio (e.g., 0.80) before the asset becomes eligible for liquidation.

A liquidation event is triggered by the protocol's smart contracts if and only if $HF_{u,t} < 1.0$.

Definition of the Event Study Window To analyze the causal factors leading to insolvency, we define a discrete event study window denoted by the interval $[T_{start}, T_{end}]$. The terminal point, T_{end}, is defined as the timestamp of the liquidation transaction block. The initial point, T_{start}, is defined as the timestamp of the last user-initiated interaction (e.g., `Deposit`, `Borrow`, or `Repay`) recorded on the ledger prior to T_{end}.

This interval relies on the assumption of *ceteris paribus* regarding asset quantities: within (T_{start}, T_{end}), the user makes no active adjustments to their portfolio. Consequently, any variation in $HF_{u,t}$ is attributable exclusively to exogenous market price fluctuations ($P_{k,t}$). This isolation allows for a deterministic reconstruction of the risk trajectory.

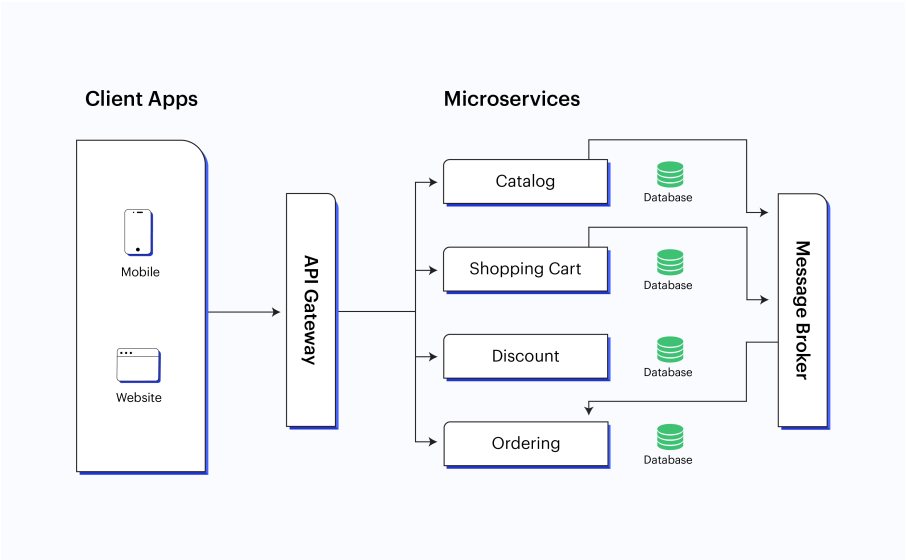


Figure 8: FastAPI Principle

Model Calibration via the Global Adjustment Factor A methodological challenge arises from the discrepancy between the continuous off-chain price data (P^{Off}) used for simulation and the discrete on-chain oracle data (P^{On}) used by the protocol to trigger liquidations. This can lead to a divergence where the simulated Health Factor at the moment of liquidation, $HF_{sim}(T_{end})$, differs slightly from the ground-truth on-chain Health Factor, $HF_{real}(T_{end})$.

To reconcile this systemic error and ensure the model converges with reality at the critical event boundary, we introduce a scalar metric termed the Global Adjustment Factor (GAF). The GAF is calculated for each subject u as:

$$GAF_u = \frac{HF_{real}(T_{end})}{HF_{sim}(T_{end})} \tag{13}$$

This factor quantifies the aggregate deviation between the two pricing sources. We then apply this scalar to calibrate the entire historical simulation trajectory:

$$HF_{adjusted}(t) = HF_{sim}(t) \cdot GAF_u \tag{14}$$

By applying this linear transformation, we preserve the morphological trend of the health factor curve derived from high-frequency market data while anchoring the final state to the immutable truth of the blockchain ledger. This ensures that the simulation accurately reflects the "distance to liquidation" as perceived by the protocol.

6.1.2 Longitudinal Health Factor Simulation

Following the extraction of the static liability structure, the study proceeded to reconstruct the dynamic risk trajectory of each subject. This involved a data fusion process wherein high-frequency, ETH-denominated price vectors retrieved from the CoinGecko API were injected into the static asset model derived from the N-1 blockchain snapshots.

For every hourly interval t within the defined window $[T_{start}, T_{end}]$, the Health Factor was iteratively computed. Since the asset quantities (Q) are invariant within this closed interval, the simulation recalculates the ratio of total collateral value to total debt value driven solely by the volatility in the external price vector (P_t).

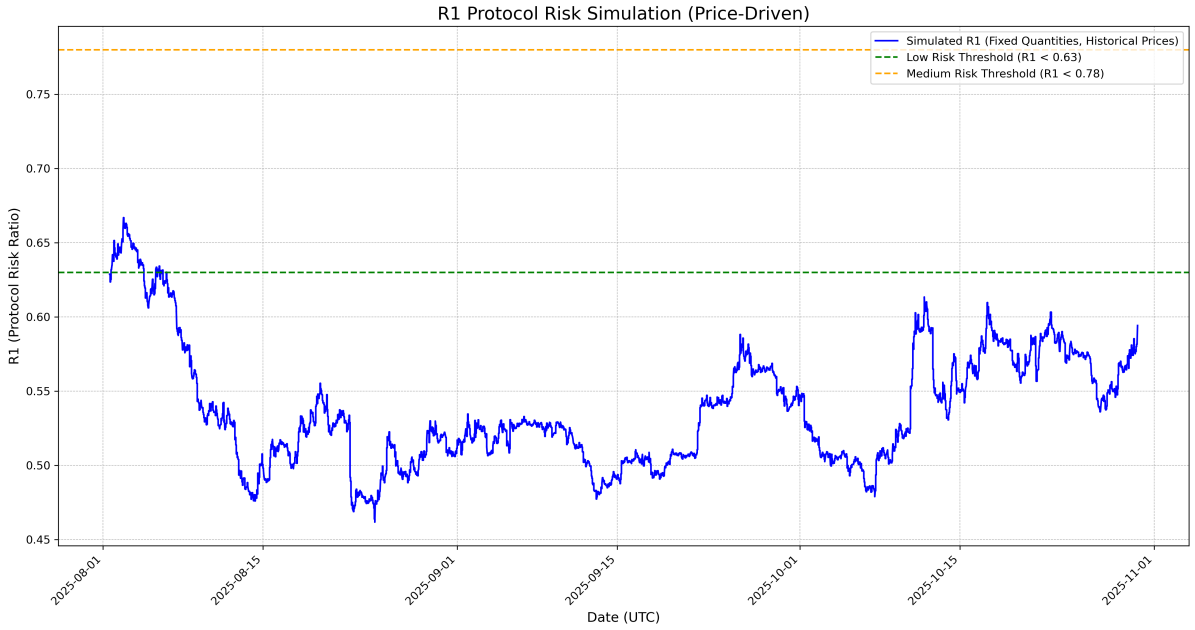


Figure 9: R1 Simulation for Target Period

To reconcile the inherent granularity differences between the continuous market data used for simulation and the discrete oracle data used by the protocol, we applied the Global Adjustment Factor (GAF) calibration method defined in the methodology. By calculating the scalar ratio between the ground-truth on-chain Health Factor and the simulated value at the terminal timestamp (T_{end}), and subsequently applying this scalar to the entire historical series, we ensured that the simulated Health Factor trajectory converges precisely with the verified ledger state at the moment of liquidation. This calibration eliminates systemic pricing errors, yielding a high-fidelity reconstruction of the solvency decay curve.

6.1.3 Dataset Classification and Validation

To rigorously validate the explanatory power of the reconstructed time series, the processed samples were stratified into two distinct categories based on the convergence of the simulation with the protocol's liquidation logic.

- **Reach (Explainable) Samples:** Defined as instances where the simulated Health Factor trajectory breached the critical insolvency threshold ($HF_{min} < 1.0$) within the study window. These samples confirm that standard market price volatility was the primary driver of the liquidation, validating the hybrid data model.
- **UnReach (Anomalous) Samples:** Defined as instances where the simulated Health Factor remained consistently above unity ($HF_t \geq 1.0$). These anomalies suggest that the liquidation was triggered by micro-structural on-chain events—such as flash loan attacks or momentary price wicks occurring within the hourly aggregation window—that are invisible to the standard resolution of off-chain market data.

The final dataset comprises a structured corpus of artifacts for each valid "Reach" subject: a JSON configuration file detailing the static liability structure and risk parame-

ters suitable for feature engineering, and a CSV file containing the normalized, calibrated Health Factor time series optimized for sequential analysis. Preliminary statistical analysis indicates a high retention rate of "Reach" samples, corroborating the efficacy of the N-1 Snapshot and GAF calibration methodologies in accurately reproducing historical solvency crises.

6.2 Simulation and Case Studies

6.2.1 Simulation Environment and Parameters

To rigorously evaluate the efficacy of the proposed Intelligent Risk Dual-Core Engine, we established a comprehensive simulation environment that mirrors the volatility of real-world DeFi markets. The simulation spans a critical observation window from **October 21, 2025, to October 30, 2025**, capturing a period of significant market fluctuations suitable for stress-testing risk management protocols.

Liquidity Stress Testing: Wallet Scenarios Recognizing that a user's ability to mitigate risk is fundamentally constrained by their available liquidity, we constructed distinct "Wallet Scenarios" to simulate varying degrees of financial pressure. These scenarios define the reserve capital available to the automated agent for debt repayment or collateral supplementation:

- **Conservative Scenario (High Liquidity Stress):** The wallet holds reserves equivalent to only **25%** of the total outstanding debt value. This scenario tests the strategy's efficiency under severe capital constraints, where full repayment is impossible.
- **Balanced Scenario (Medium Liquidity Stress):** The wallet holds reserves equivalent to **75%** of the total debt. This represents a typical user who can partially deleverage but cannot fully unwind their position.
- **Aggressive Scenario (Low Liquidity Stress):** The wallet holds reserves equivalent to **110%** of the total debt (or 100% of collateral value). This simulates a well-capitalized user capable of completely liquidating their debt if necessary, allowing the strategy to prioritize optimality over survival.

Key Risk Parameters The decision-making logic of the engine relies on a set of statistically derived thresholds and temporal parameters, calibrated to balance sensitivity with robustness:

- **Systemic Risk Threshold (R_I):** Set at **0.63**. This value serves as the boundary for identifying a fragile macro-environment, derived from the median historical distribution of DeFi systemic risk indicators.
- **Health Factor Velocity Threshold (HF_{vel}):** Defined as the bottom **20th percentile (q20)** of historical HF changes. This dynamic threshold allows the system to detect rapid deterioration in solvency before the absolute HF breaches critical levels.

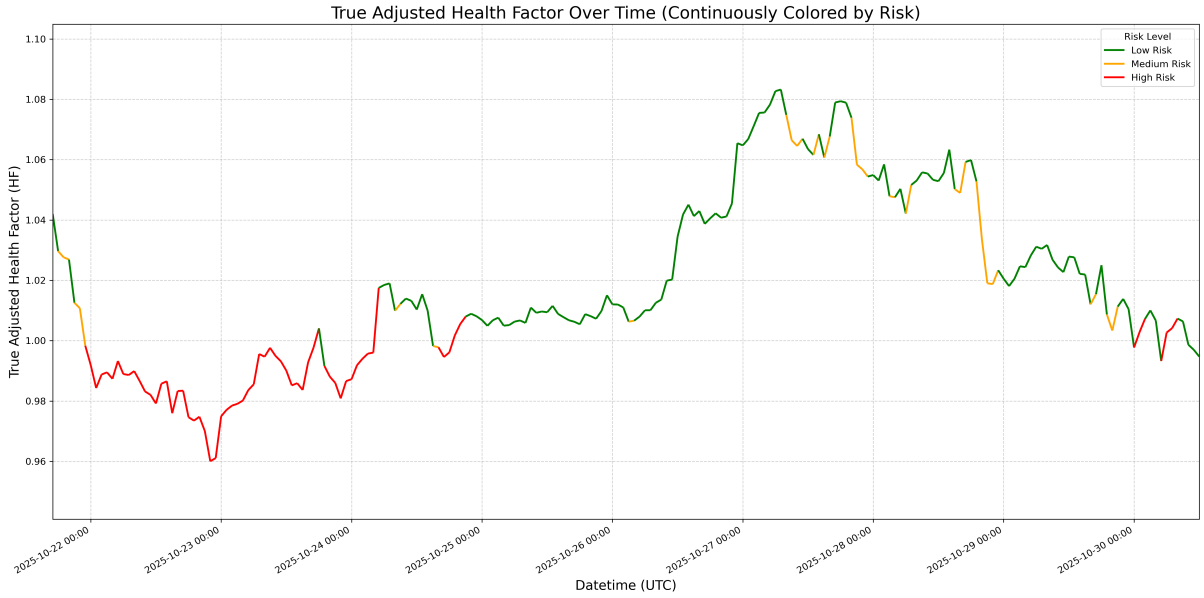


Figure 10: Phase Identification for Target Address

- **Static Health Factor Threshold (HF_{static}):** Set at the bottom **30th percentile (q30)** of the weighted predicted HF, serving as a baseline for the "Medium Risk" trend warning.
- **Liquidation Probability Horizon (T):** Fixed at **6 hours**. The Geometric Brownian Motion (GBM) model calculates the probability of the user's HF dropping below 1.0 within this forward-looking window, providing a proactive metric for risk quantification (Metric 2).

6.2.2 Risk Phase Identification and Triggering

The core of the Dual-Core Engine is its "Judge Phase," a sentinel mechanism designed to classify the user's risk state in real-time. Unlike traditional models that rely on a single static threshold (e.g., $HF < 1.05$), our system employs a multi-dimensional risk matrix that synthesizes macro-systemic conditions with micro-idiosyncratic positioning.

Multi-Dimensional Risk Metrics The identification module continuously monitors three distinct vectors to detect anomalies:

- **Systemic Fragility (R_I):** Quantifies the macro volatility of the lending protocol. A value exceeding the historical median of **0.63** indicates a high probability of cascading liquidations.
- **Long-Term Solvency Trend ($HF_{weighted}$):** A time-weighted average of predicted Health Factors over future steps ($t+1$ to $t+3$). This filters out short-term market noise to reveal the underlying solvency direction.
- **Instantaneous Velocity (HF_{vel}):** Measures the rate of change in the user's position ($\Delta HF/\Delta t$). A drop falling into the bottom **20th percentile** of historical data signals a "flash crash" scenario that requires immediate attention, even if the absolute HF remains high.

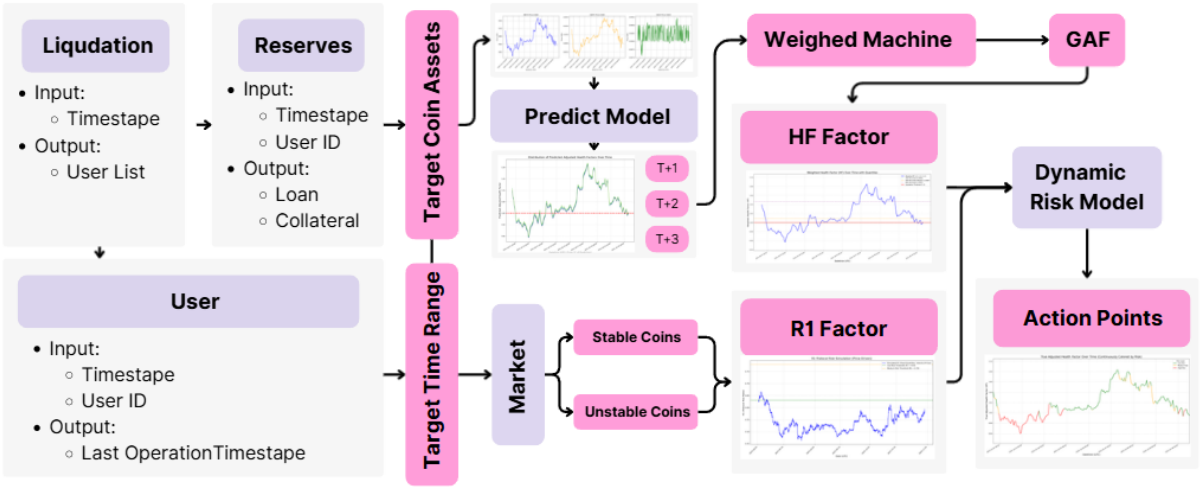


Figure 11: How to Find the Action Point

Triggering Logic and State Transition The system dynamically assigns a risk state based on the convergence of these metrics. A transition from **Low Risk** to **Medium Risk** acts as the primary trigger for the simulation engine. This state is activated if *any single* warning threshold is breached, ensuring a conservative approach to risk. Conversely, a transition to **High Risk** requires the confluence of multiple severe factors (e.g., High Systemic Risk *and* High Velocity Drop) or an imminent liquidation signal ($HF < 1.0$).

Velocity-Driven Trigger The necessity of this dynamic approach is exemplified by the simulation event on **October 21, 2025, at 18:00 UTC**.

- **State:** The user's static Health Factor was **1.0296**, a level traditionally considered "safe" and above the immediate liquidation threshold.
- **Trigger:** However, the Judge Phase detected a significant negative velocity in the user's position (classified as `HF_Dropping_Only`).
- **Outcome:** Despite the safe static level, the system correctly identified this rapid deterioration as a pre-crisis signal. It escalated the status to **Medium Risk**, triggering the strategy engine. Subsequent analysis revealed a non-zero liquidation probability of **2.89%** within the next 6 hours, validating the decision to intervene preemptively before the HF breached the critical 1.0 line.

6.2.3 Case Study I: Comparative Strategy under Medium Risk

To illustrate the engine's decision-making process during the "Medium Risk" phase, we analyze a representative simulation event occurring on **October 21, 2025, at 18:00 UTC**. This scenario highlights the critical trade-off between capital efficiency (operation cost) and solvency safety (Health Factor improvement).

Baseline State and Trigger At this timestamp, the user, modeled with a "Conservative" wallet profile (holding only $510.14 in reserves), exhibited a Health Factor of **1.0296**. While technically above the liquidation threshold of 1.0, the Judge Phase

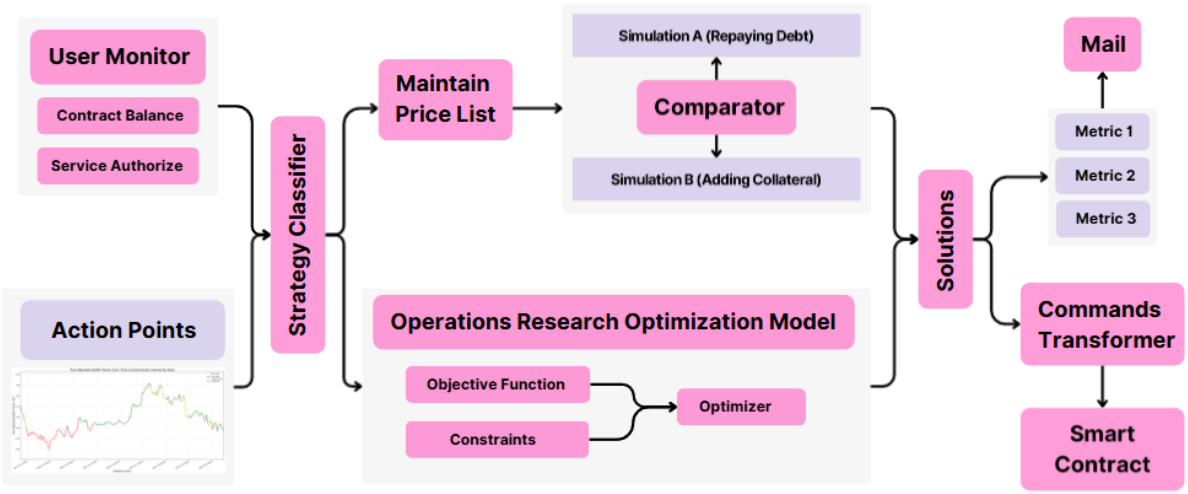


Figure 12: How to Act

flagged a critical anomaly: the position's Health Factor velocity was negative, indicating a rapid deterioration. Consequently, the system calculated a non-negligible liquidation probability of **2.89%** over the next 6 hours, triggering a *Medium Risk* alert.

Simulation of Countermeasures The Strategy A engine simulated two mutually exclusive defensive actions using the available $510.14 budget:

- **Action A: Debt Repayment (USDC):** The agent simulated using the entire reserve to repay USDC debt.
  - **Solvency Impact:** This action significantly reduced the denominator of the HF equation, resulting in a substantial HF increase of **+0.3432**, raising the final HF to **1.3728**.
  - **Economic Impact:** Due to unfavorable short-term price fluctuations in stablecoins, this operation incurred a slight slippage cost of **-$0.048**.
- **Action B: Collateral Supplementation (WETH):** Alternatively, the agent simulated purchasing WETH to deposit as collateral.
  - **Solvency Impact:** This increased the numerator of the HF equation but yielded a more modest HF increase of **+0.2021**, raising the final HF to **1.2317**.
  - **Economic Impact:** Conversely, purchasing WETH at this specific moment would have resulted in a positive execution gain (negative cost) of **+$6.10** due to favorable intraday price movements.

Decision Logic and Outcome The engine compared both outcomes against the success criteria ($HF_{after} > HF_{threshold}$). While both actions successfully stabilized the position, the system prioritized **risk minimization** over short-term economic gain. Despite Action B offering a financial incentive (execution gain), Action A provided a superior safety buffer against future volatility (a 34% improvement vs. 20%). Consequently, the decision module executed **Action A (Repay Debt)**, successfully reducing the 6-hour liquidation probability to **0%**.

6.2.4 Case Study II: Optimization under High Risk

While Strategy A suffices for moderate threats, "High Risk" scenarios—characterized by imminent liquidation ($HF < 1.0$) or extreme systemic fragility—require a more aggressive and mathematically optimal response. We analyze the performance of Strategy B using a critical event observed on **October 24, 2025, at 16:00 UTC**.

Crisis Scenario At this specific timestamp, the simulated user, holding an "Aggressive" wallet with **$2,244.62** in reserves, faced a solvency crisis. The agent's Health Factor had deteriorated to **0.9978**, breaching the liquidation threshold of 1.0. Consequently, the *Judge Phase* calculated a liquidation probability (P_{liq}) of **100%**, triggering an immediate escalation to the High Risk state. Unlike Strategy A, which merely seeks to cross a safety line, Strategy B was tasked with deploying the capital to maximize the "Net Monetary Value" (Z) of the recovery.

Optimization Process The optimization engine executed a grid search over the decision space ($V_{repay}, V_{add\_WETH}, V_{add\_cbBTC}$), evaluating 66 distinct allocation combinations with a 10% granularity. The objective function was defined as:

$$\max Z = \lambda \cdot (HF_{after} - HF_{before}) - Cost_{total} \tag{15}$$

where $\lambda = 100$, quantifying the monetary utility of solvency improvement.

Optimal Solution and Execution The model identified a global maximum for Z at a score of **10,836.47**. The optimal capital allocation vector was:

- **90% ($2,020.16)** allocated to repaying USDC debt.
- **10% ($224.46)** allocated to supplementing WETH collateral.
- **0%** allocated to cbBTC.

This "hybrid" approach proved superior to singular actions. By aggressively reducing the debt denominator (USDC) while simultaneously bolstering the collateral numerator (WETH), the strategy exploited the Health Factor's non-linear sensitivity to leverage reduction.

Result Verification The execution of this optimal bundle yielded decisive results:

- **Solvency Restoration:** The Health Factor skyrocketed from **0.9978** to a hyper-safe level (approximately **109.0**), driven by the near-total elimination of debt.
- **Risk Elimination:** The 6-hour liquidation probability was reduced from **100%** to **0%**, completely neutralizing the immediate threat.
- **Capital Efficiency:** The strategy incurred a negligible operation cost while securing the highest possible safety margin per dollar spent, validating the utility of the NMV optimization model in crisis management.

7 On-chain and Off-chain Interaction Prototype

7.1 Prototype Design

7.1.1 System Architecture: The Bridge Paradigm

To overcome the inherent computational constraints and high gas costs associated with complex mathematical modeling on the Ethereum Virtual Machine (EVM), we propose a hybrid interaction architecture termed the "Bridge Paradigm." This framework decouples high-frequency risk computation from on-chain asset settlement, establishing a secure link between an *Off-chain Decision Layer* and an *On-chain Execution Layer*.

The Off-chain Decision Layer This layer serves as the computational brain of the system. Hosted in a local environment, it is responsible for:

- **Data Ingestion:** Aggregating real-time price feeds and on-chain Health Factors.
- **Model Execution:** Running the computationally intensive algorithms defined in Strategy A (Comparative Simulation) and Strategy B (Net Monetary Value Optimization).
- **Instruction Generation:** Converting the optimal capital allocation vector derived from the risk models into precise transaction payloads.

By performing these operations off-chain, the system bypasses EVM execution limits, allowing for sophisticated grid search optimizations that would be economically infeasible to run directly on a smart contract.

The On-chain Execution Layer The on-chain component is instantiated as the `Executor.sol` smart contract. Designed with a minimalist philosophy, this contract acts as a semi-custodial vault. Unlike traditional DeFi agents that calculate risk internally, the `Executor` is logic-agnostic; its sole function is to verify the cryptographic signature of the incoming transaction and execute the authorized fund transfer atomically. This separation ensures that while the decision logic is flexible and complex, the execution remains transparent, verifiable, and immutable on the blockchain.

7.1.2 On-chain Infrastructure: The Executor Contract

The on-chain execution module is instantiated as the `Executor` smart contract, developed in Solidity. This contract is designed to function as a secure, minimalist gateway for capital deployment, strictly governed by cryptographic ownership protocols to mitigate unauthorized access and ensure atomic execution of risk management decisions.

Ownership and Access Control To establish a robust chain of command, the contract integrates the standard `Ownable` module from the OpenZeppelin library. Upon deployment, administrative privileges are cryptographically assigned to the deploying address (Wallet A), which corresponds to the secure private key managed by the off-chain decision agent. The `onlyOwner` modifier is rigorously applied to all critical functions, ensuring that the contract rejects any external calls not signed by the authorized decision-making entity. This mechanism effectively insulates the on-chain funds from unauthorized actors.

Atomic Execution Mechanism The core operational logic resides in the `executeTransfer` function. This method accepts a target address and a specific monetary value as parameters, derived directly from the off-chain optimization results. Internally, the contract utilizes the low-level `.call` primitive to perform the transfer of the native currency (Ether). This approach ensures atomicity, meaning the transaction either completes successfully or reverts entirely, preserving the integrity of the ledger. Prior to execution, the function enforces validation checks to ensure the target address is valid and that the contract maintains sufficient liquidity to fulfill the request.

Security and Liquidity Management To ensure the safety and recoverability of assets, the contract implements a secure `withdraw` function. This emergency mechanism allows the owner to retrieve the entire residual balance of the contract, facilitating liquidity rebalancing or emergency shutdowns. By restricting these capabilities exclusively to the owner via the `payable` constructor initialization, the system maintains a high degree of security while retaining the flexibility required for dynamic asset allocation.

7.1.3 Off-chain Oracle and Trigger Mechanism

The bridge between the analytical engine and the blockchain ledger is established via a dedicated Python automation script, `trigger_transfer.py`. acting as an ad-hoc Oracle. This component translates the optimized capital allocation decisions into valid Ethereum transactions, managing the complexities of network communication and cryptographic security.

Secure Environment Configuration To maintain strict operational security (OpSec), the system avoids hardcoding sensitive credentials directly into the source code. Instead, it employs the `dotenv` library to load configuration parameters from a local `.env` file. Critical variables, including the `SEPOLIA_RPC_URL` for network access and the `WALLET_A_PRIVATE_KEY` for transaction signing, are injected into the runtime environment only at execution. This practice ensures that cryptographic keys remain isolated from the logic layer, reducing the attack surface for key exfiltration.

EIP-1559 Transaction Construction The system constructs transactions in compliance with the Ethereum Improvement Proposal 1559 (EIP-1559) standard to ensure predictable gas fee estimation and timely inclusion in blocks. The transaction payload is assembled as a dictionary containing:

- **Chain ID:** To prevent replay attacks across different networks (e.g., Sepolia Testnet).
- **Gas Parameters:** Explicitly defining `maxFeePerGas` and `maxPriorityFeePerGas` to optimize transaction costs based on current network congestion.
- **Calldata:** The encoded function call to `executeTransfer`, encapsulating the target address and the precise fund amount derived from the optimization model.

Cryptographic Signing and Broadcasting Once constructed, the transaction object undergoes offline signing using the `web3.py` library. The local script utilizes the loaded private key to generate a purely cryptographic signature, producing a `raw_transaction`

byte string. This signed payload is then broadcast to the Ethereum network via the Remote Procedure Call (RPC) node using `eth_sendRawTransaction`. This offline signing paradigm ensures that the private key never leaves the secure local environment, maintaining the non-custodial integrity of the system while enabling autonomous on-chain execution.

7.1.4 Interaction Workflow and Verification

To demonstrate the end-to-end validity of the proposed dual-layer architecture, we delineate the complete lifecycle of a risk management operation, tracing the data flow from the off-chain optimization engine to the immutable on-chain ledger.

The Execution Lifecycle The interaction process follows a rigorous four-stage sequence:

1. **Decision Synthesis (Off-chain):** The process initiates with the Python-based optimization engine (Strategy B). Upon identifying a "High Risk" signal ($P_{liq} \approx 100\%$), the model executes a grid search to determine the optimal capital allocation vector. For instance, it may generate a decision to allocate **90%** of reserves to debt repayment. This output is serialized into a precise numerical value (in Wei) and passed to the trigger module.

2. **Transaction Construction and Broadcast:** The `trigger_transfer.py` script acts as the bridge. It initializes a Web3 connection to the Sepolia testnet and loads the `Executor` contract interface using its ABI. The script then constructs a raw transaction invoking the `executeTransfer` function, populating the payload with the calculated optimal amount and the target protocol address. This transaction is cryptographically signed offline using the administrator's private key and broadcast to the network via the RPC node.

3. **Consensus and Settlement (On-chain):** Once broadcast, the transaction propagates through the Ethereum mempool. Validators include the transaction in a block, executing the state change on the `Executor` contract. The Python script utilizes the `wait_for_transaction_receipt` method to synchronously monitor the network, pausing execution until the transaction is confirmed by the consensus protocol.

4. **Cryptographic Verification:** Upon confirmation, the script retrieves the transaction receipt, verifying the execution status (`status: 1` indicating success). The returned **Transaction Hash** (e.g., `0x...`) serves as an immutable audit trail. By querying this hash on a block explorer (e.g., Etherscan), we can independently verify the gas consumption, the exact value transferred, and the block timestamp, proving that the off-chain risk decision was successfully translated into a finalized on-chain action.

7.2 Prototype Deployment

The prototype deployment focused on establishing a robust, validated communication link between the off-chain **Intelligent Risk Dual-Core Engine** and the **Aave V3 Protocol**

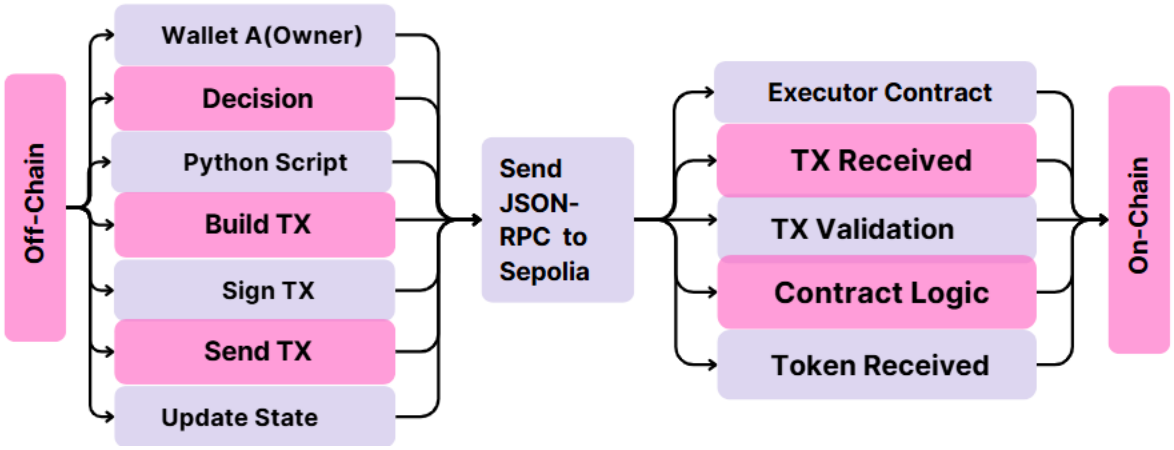


Figure 13: From Off-Chain to On-Chain

on the Sepolia test network. This setup was critical for demonstrating the final step of the risk management strategy: the automated, on-chain execution of portfolio adjustments.

7.2.1 Technical Stack and Environment

The execution environment utilized a modern, asynchronous Web3 stack to ensure reliable and efficient transaction handling:

- **Network:** Sepolia Test Network (Ethereum)
- **Tooling:** Hardhat for environment management and local testing.
- **Ethereum Client: Viem**, chosen for its modularity and robust support for modern TypeScript applications, was used to construct both the `publicClient` (for reading on-chain data and checking transaction status) and the `walletClient` (for signing and submitting transactions).
- **Aave Contracts:** The interaction leveraged the official Aave V3 contract ABIs for the core **Pool** (the main lending/borrowing logic) and the **WrappedTokenGatewayV3** (the required helper contract for handling native ETH).

7.2.2 On-Chain Transaction Flow Validation

The primary goal of the prototype was to validate the four core transactions required by the Portfolio Adjustment Strategy. All interactions were designed to use a single designated wallet, the `account`, which served as the executor for the autonomous assistant.

1. **Depositing Collateral (ETH):**

- **Purpose:** To increase the position's Health Factor (**HF**) by adding native ETH as collateral.
- **Mechanism:** The `WrappedTokenGatewayV3.depositETH` function was used. This function accepts native ETH in the transaction's `value` field, atomically wrapping it into WETH and depositing the WETH into the Aave Pool as collateral in a single call. This is the required mechanism for native token deposits.

- **Validation:** A successful transaction confirmed the increase in the user's aWETH balance and a corresponding increase in the overall Health Factor.

2. Borrowing Assets (e.g., USDT):

- **Purpose:** The strategy may recommend borrowing to optimize capital efficiency after a period of stability or low-risk identification.
- **Mechanism:** The `Pool.borrow` function was called, specifying the target asset address (e.g., Sepolia USDT), the desired amount, the interest rate mode (Variable or Stable), and the account acting on behalf of the user.
- **Validation:** A successful transaction resulted in the borrowed asset being transferred to the executor's wallet and the user's HF being lowered, confirming the creation of new debt.

3. Repaying Debt (ERC20):

- **Purpose:** To mitigate risk by reducing debt, typically triggered by a medium or high-risk identification.
- **Mechanism:** Repayment of an ERC20 asset (like USDT) requires a two-step process:
  - **Step 3a: Approval:** The executor first calls `approve` on the ERC20 asset contract to grant the Aave Pool a spending allowance for the repayment amount.
  - **Step 3b: Repay:** The executor then calls `Pool.repay`, which allows the Pool to pull the approved tokens from the executor's wallet to settle the outstanding debt.
- **Validation:** Successful execution confirmed the reduction of the user's stablecoin debt and the resultant increase in the Health Factor.

4. Withdrawing Collateral (ETH):

- **Purpose:** To free up excess collateral when the HF is safely above liquidation thresholds, optimizing capital utility.
- **Mechanism:** Similar to the deposit, the withdrawal of native ETH is handled by the `WrappedTokenGatewayV3.withdrawETH` function. This function unwraps the WETH and sends the resulting native ETH back to the executor.
- **Validation:** A successful transaction confirmed the decrease in aWETH holdings and the transfer of native ETH to the wallet.

This structured validation of all four core interactions confirms that the Portfolio Adjustment Assistant can reliably and autonomously execute the necessary on-chain operations to manage liquidation risk.

8 User Interface Implementation

To bridge the gap between theoretical modeling and practical application in virtual banking, we developed a high-fidelity prototype of the *Intelligent Risk Dual-Core Engine*. This visualization layer is designed to translate complex underlying algorithmic outputs—such as the Systemic Risk Indicator (R_I), LSTM-based Health Factor predictions, and portfolio optimization solutions—into actionable insights for risk managers. The interface design strictly follows the "Macro-Micro-Action" hierarchical logic proposed in our methodology.

8.1 Macro-Prudential Monitoring

The systemic view, illustrated in Fig. 14, serves as the centralized command center. It visualizes the protocol-wide fragility using a gauge chart calibrated to the R_I threshold derived in Section III. This dashboard enables risk officers to identify "Fragile States" (e.g., $R_I > 0.63$) at a glance, aggregating real-time data on Total Value Locked (TVL) and active liquidation events across the liquidity pool.

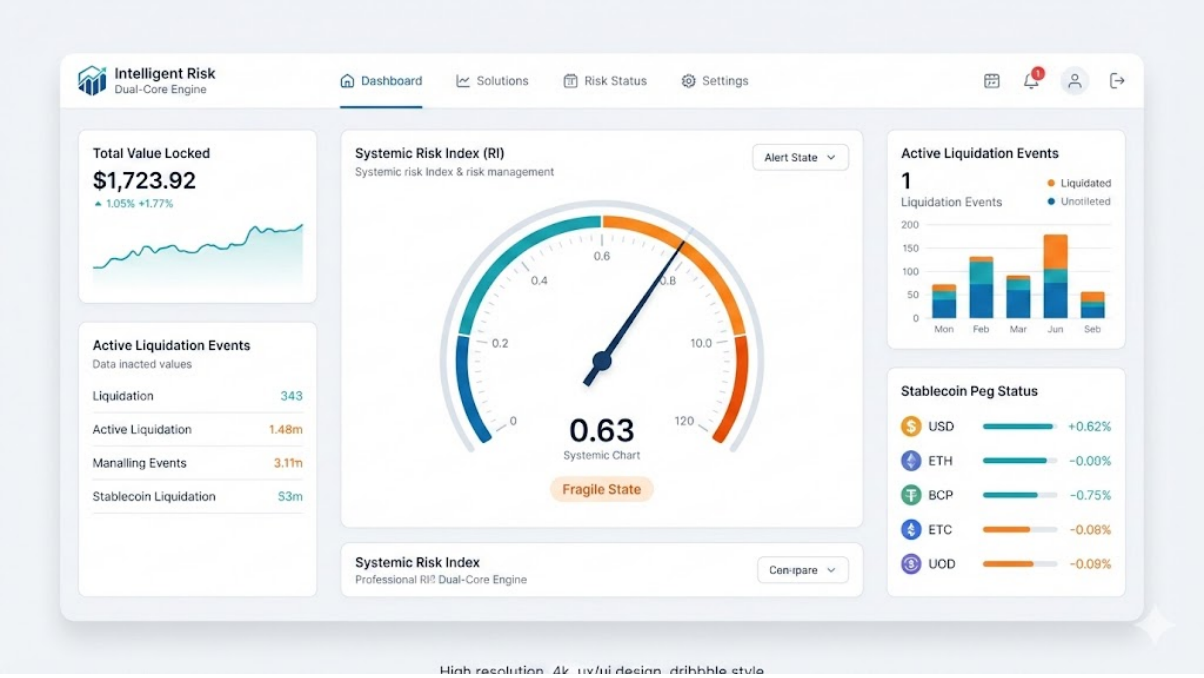


Figure 14: Macro-Prudential Systemic Risk Monitoring Dashboard

8.2 Micro-Level Predictive Surveillance

Drilling down to the granular level, Fig. 15 demonstrates the individual position monitoring module. Unlike traditional interfaces that display only static snapshots, this module integrates the temporal dimension. It visualizes the *Health Factor Velocity* and overlays the LSTM-predicted trajectory (dotted line) against the liquidation threshold. This design allows for the early detection of "High Velocity" distress signals before actual insolvency occurs.

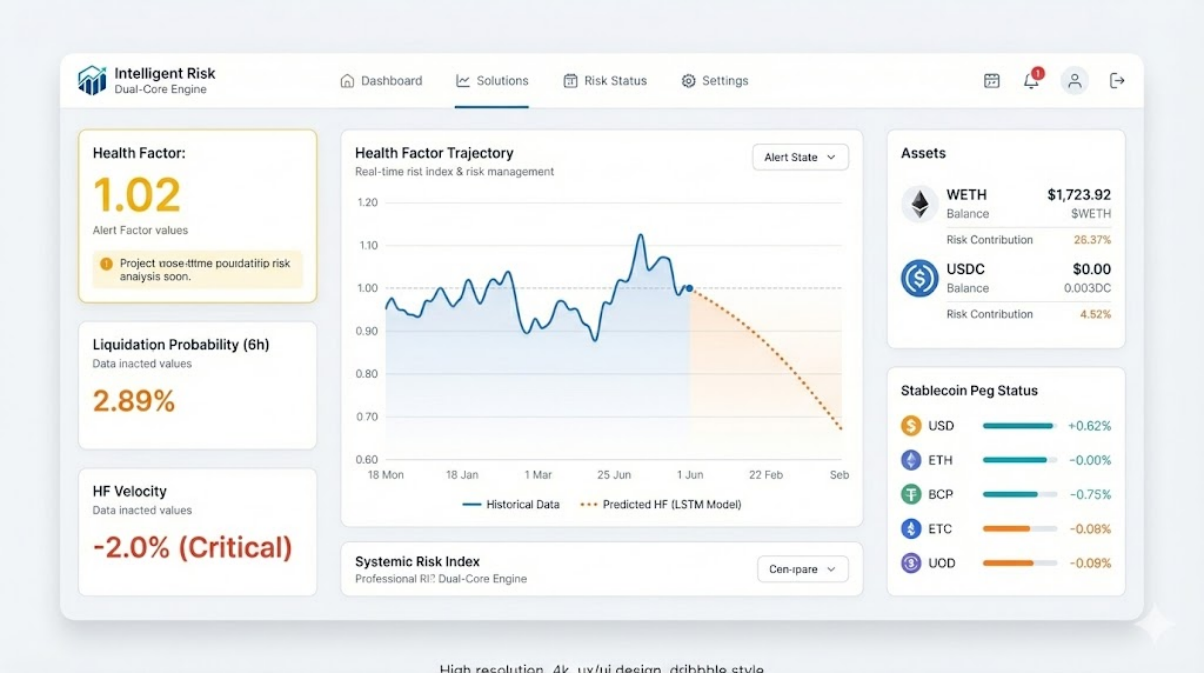


Figure 15: Micro-Level Individual Risk Assessment Interface

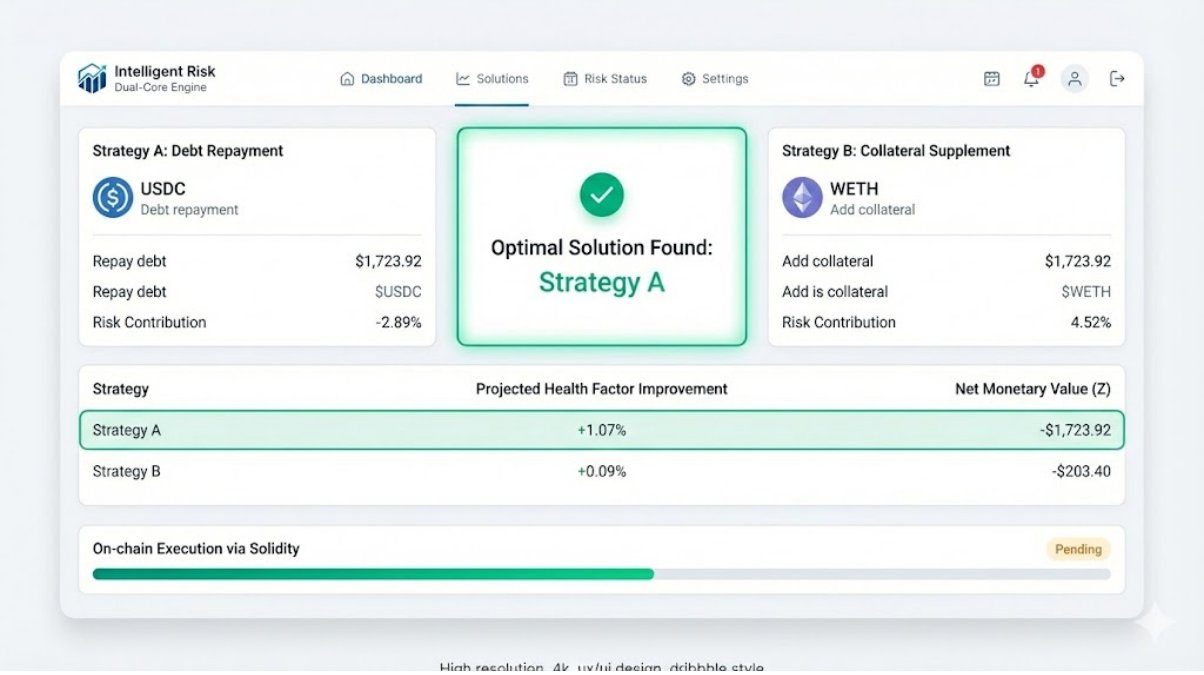


Figure 16: Algorithmic Strategy Optimization and Execution Console

8.3 Strategic Optimization and Execution

Upon triggering a risk alert, the system transitions to the decision support phase shown in Fig. 16. This console visualizes the output of the Net Monetary Value (Z) optimization model. It provides a comparative analysis between different mitigation strategies (e.g., Debt Repayment vs. Collateral Supplementation), explicitly displaying the optimal solution that maximizes capital efficiency. The integration of an "On-chain Execution" status bar emphasizes the system's capability to bridge off-chain computation with atomic blockchain settlement.

9 Conclusion

This study successfully engineered and validated the "Intelligent Risk Dual-Core Engine," a comprehensive automated framework designed to mitigate liquidation risks within the Aave protocol. By synthesizing macro-prudential systemic indicators (R_I) with micro-level predictive analytics (LSTM-based price forecasting), we have transitioned DeFi risk management from a passive, reactive paradigm to a proactive, dynamic one.

9.1 Research Contributions and System Advantages

The primary contribution of this work lies in the integration of off-chain intelligence with on-chain atomic execution. Our simulations demonstrate that the proposed multi-factor risk identification model significantly outperforms traditional static threshold mechanisms. By incorporating the Health Factor velocity (`HF_pct_change`) and the Systemic Risk Indicator (R_I), the system successfully identifies "flash crash" scenarios that static models often miss. Furthermore, the implementation of the "Bridge Paradigm" architecture effectively solves the computational bottleneck of smart contracts. It allows for the execution of complex Net Monetary Value (NMV) optimization algorithms off-chain while maintaining the security and immutability of on-chain asset settlement through the `Executor` contract.

9.2 Limitations and Constraints

Despite its demonstrated efficacy, the current prototype operates under specific constraints that warrant critical attention:

- **Centralization Risk in Decision Layer:** While the execution layer is decentralized and trustless, the decision layer currently relies on a local Python-based agent. This introduces a single point of failure; if the off-chain server experiences downtime or connectivity issues during a market crisis, the protective mechanism fails.
- **Economic Viability under Congestion:** The simulation highlighted the friction imposed by Gas Fees. In scenarios of extreme network congestion—which often correlate with market crashes—the transaction costs for "Medium Risk" adjustments may outweigh the economic benefits of debt repayment for smaller portfolios, creating an "economic protection floor."

- **Model Dependency on Historical Patterns:** The LSTM model, while accurate in backtesting ($R^2 > 0.98$), operates on the assumption that future volatility mirrors historical data. In unprecedented "Black Swan" events with structural market breaks, the predictive power may diminish.

9.3 Future Outlook

Future iterations of this research should focus on three strategic directions to enhance robustness and scalability:

1. **Decentralization of the Compute Layer:** Transitioning the off-chain Python agent to a decentralized compute network (such as Chainlink Functions or Zero-Knowledge Machine Learning, ZK-ML) would eliminate the single point of failure and ensure true trustlessness.

2. **Layer 2 Deployment:** Migrating the execution module to low-cost Layer 2 solutions (e.g., Arbitrum or Optimism) would drastically reduce gas costs, making high-frequency portfolio adjustments economically viable for a broader range of investors.

3. **Cross-Protocol Composability:** Expanding the strategy to aggregate liquidity across multiple lending protocols (e.g., Compound, Morpho) could allow for "Flash Repayment" strategies, where the system borrows from a lower-risk protocol to repay debt on a high-risk protocol atomically.

In conclusion, while challenges regarding centralization and cost remain, this study establishes a foundational architecture for the next generation of DeFi risk management—one that is intelligent, autonomous, and mathematically optimized.

Author Contributions

Guoxuan Sun: Led the Conceptualization and Project Administration, specifically focusing on data curation, the formulation of portfolio rebalancing strategies, and the architectural construction of the on-chain/off-chain interaction prototype. Oversaw the entire project progression, playing a lead role in topic definition, manuscript drafting, and the final presentation.
Chenwei Xu: Contributed to Methodology and Validation, specifically focusing on the construction, optimization, and backtesting of cryptocurrency price prediction models. Led the development of forecasting architectures, hyperparameter tuning, and performance evaluation across multiple market scenarios. Participated throughout the project lifecycle, supporting technical analysis, empirical testing, manuscript drafting, and the final presentation.
Siyu Huang: Contributed to Strategy Design and Methodology, specifically focusing on the development of the Portfolio Adjustment Strategy. Took the lead in designing the Multi-Factor Dynamic Risk Identification Model, formulating the portfolio adjustment operation model, and designing tiered execution strategies. Responsible for the design of evaluation metrics.
Yexin Li: Contributed to Conceptualization and Methodology, specifically focusing on the design of temporal triggering mechanisms for risk identification and the formulation

of portfolio adjustment protocols. Participated in the full project lifecycle, playing a key role in topic selection, project scheduling, manuscript drafting, and final presentation.
Jialiang Zhou: Engineered and validated the core Aave V3 DeFi interaction prototype on Sepolia. This included the complete technical implementation of transaction lifecycle: depositing and withdrawing ETH collateral using the WETH Gateway, executing asset borrowing, and managing the two-step ERC20 debt repayment. This work established the reliable, functional smart contract interface for the project's strategic models.