# Project Report: GenAI Advisor, based on the Financial Advisor Bot template

Andreas Mallios
BSc Computer Science (ML and AI)
Student number: 200128357

June 2025

# Contents


**1 Introduction** **3**
- 1.1 Concept . . . 3
- 1.2 Motivation . . . 4
- 1.3 Scope . . . 4

**2 Literature Review** **6**
- 2.1 Generative AI in Financial Forecasting . . . 6
- 2.2 Explainable AI in Financial Systems . . . 6
- 2.3 Rule-based vs Machine Learning Investment Strategies . . . 7
- 2.4 Designing for Non-Technical Users in FinTech . . . 7
- 2.5 Backtesting and Evaluation in FinTech . . . 8
- 2.6 Synthesis and Research Gap . . . 8

**3 System Design** **10**
- 3.1 System Overview & Design Rationale . . . 10
- 3.2 Data Pipeline . . . 11
- 3.3 Strategy Engine . . . 12
- 3.4 Explanation Generator . . . 13
- 3.5 User Interface . . . 13
- 3.6 Backtesting & Evaluation . . . 14

**4 Implementation** **16**
- 4.1 Overview . . . 16
- 4.2 System Architecture . . . 16
- 4.3 Development Process . . . 16
- 4.4 Key Modules and Features . . . 16
  - 4.4.1 Data Ingestion . . . 16
  - 4.4.2 Strategy Engine . . . 17
  - 4.4.3 Explanation Generator . . . 17
  - 4.4.4 Interface . . . 18
- 4.5 Implementation Challenges . . . 18

**5 Evaluation** **20**
  - 5.0.1 Testing and Evaluation Support . . . 20

**6 Conslusion** **22**
- 6.1 Summary and Forward Plan . . . 22

# 1 Introduction

Artificial Intelligence (AI) continues to reshape global industries, with Generative AI (GenAI) standing out as one of the most influential recent developments. GenAI models, such as large language models (LLMs) and diffusion architectures, have driven widespread adoption and significant investment, particularly in the technology sector. Yet, for retail investors, the tools available to access this growth remain largely generic and inaccessible, both in terms of domain focus and interpretability.

Private equity and venture capital investments in GenAI startups are becoming larger and more targeted as investor strategies evolve with the technology. Funding in GenAI exceeded $56 billion in 2024, nearly double the $29 billion recorded in 2023 (14).

As shown in Figure 1 below, while the number of funding rounds declined in 2024, the overall deal value surged. This suggests a consolidation of capital into larger, infrastructure-oriented bets, aligning with investor confidence in scalable GenAI deployments. These trends highlight the need for tools that help investors evaluate publicly listed companies best positioned to benefit from this structural shift.

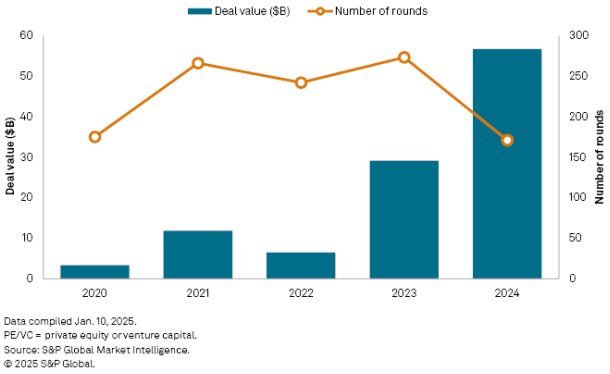


Figure 1: PE/VC investments in GenAI, 2020–2024. Source: S&P Global Market Intelligence (14).

This project responds to that gap by proposing the **GenAI Advisor**, a system that delivers explainable stock recommendations specifically for companies driving or enabling the GenAI revolution. Based on the *Financial Advisor Bot* template, the system is designed for non-technical users, providing accessible, transparent investment signals that go beyond traditional model outputs.

## 1.1 Concept

The GenAI Advisor is conceived as a personalised financial guidance tool for individual investors interested in the GenAI sector. It provides interpretable recommendations, whether to buy, hold, or sell specific stocks, alongside human-readable explanations for these decisions. Unlike traditional investment tools that obscure decision logic behind opaque algorithms or technical jargon, this system foregrounds explainability and domain relevance.

The intended users of this system are individual retail investors interested in emerging technology themes but lacking the technical background to interpret raw financial signals or operate sophisticated trading tools. These users are typically non-specialists with limited exposure to algorithmic finance or machine learning models. For this group, the GenAI Advisor offers an accessible tool to support their decisions, that reduces complexity while preserving analytical rigour. It is also suitable for students, educators, or early-stage investors exploring explainable AI in financial contexts.

At its core, the GenAI Advisor is built on the belief that financial tools should be not only accurate but also comprehensible. This is particularly important in thematic investing, where

users are drawn not only by quantitative performance but by interest in specific technological trends. In this context, the system acts as a digital intermediary between complex market signals and users' intuitive understanding, offering advice that is grounded and traceable, with a clearly scoped set of companies.

The conceptual novelty lies in its hybrid orientation; it combines evidence-driven financial analysis with a commitment to interpretability. The Advisor does not aim to outperform the market through high-frequency trading (HFT) or proprietary forecasting. Instead, it prioritises accessibility, domain focus, and trustworthiness, delivering insights users can understand and apply in their decision-making. In doing so, it invites users into the analytical process rather than substituting for it.

Furthermore, the GenAI Advisor embraces thematic integrity by carefully curating its stock universe. It avoids general-purpose technology firms in favour of those with demonstrable involvement in the GenAI ecosystem, whether through infrastructure, tooling, model development, or deployment. The rationale for company inclusion and exclusion is detailed in the project scope (Section 1.3). This conceptual grounding ensures that recommendations are not only technically derived but also strategically aligned with the user's thematic intent.

## 1.2 Motivation

The motivation for this project is both technical and user-oriented. GenAI represents an emerging thematic investment opportunity that remains underserved by existing tools. Simultaneously, there is increasing demand for transparent and accessible AI systems tailored to the needs of retail investors in the financial domain.

Traditional robo-advisors, such as Nutmeg and Moneyfarm, provide automated portfolio allocation but are domain-agnostic and opaque in how decisions are made. While open-source trading bots like Freqtrade offer flexibility, they require technical expertise and lack natural language interfaces. StockGPT (9) demonstrates the potential of generative models in stock prediction, while recent work on XAI-integrated forecasting (10) highlights the importance of trust and interpretability in financial AI systems.

By combining simple rule-based logic with predictive models and natural language explanations, the GenAI Advisor aims to deliver recommendations that are not only accurate but also intelligible. A local LLM serves as an interpretive layer rather than a decision engine, ensuring that outputs remain grounded in logic and statistical inference, while still being comprehensible to the user.

Beyond usability, the project contributes to responsible AI by ensuring data privacy and reproducibility. No personal or sensitive data is collected, and all sources are public. The LLM is hosted locally to avoid transmitting user inputs externally. The system is explicitly intended for educational and demonstrative use, and includes appropriate disclaimers to distinguish it from regulated financial services.

## 1.3 Scope

The GenAI Advisor focuses on a curated set of publicly traded companies whose business models are materially driven by Generative AI. The portfolio includes firms across three categories: (1) software developers and strategic investors (e.g., Microsoft, Alphabet, Meta, Amazon); (2) hardware providers (e.g., Nvidia, AMD, Intel, Marvell); and (3) infrastructure suppliers (e.g., TSMC, SK hynix, Broadcom, Arista Networks). These companies were selected for their direct contributions to GenAI model development, deployment, or enablement.

To maintain thematic focus and ensure data availability, the system excludes private companies, indirect holdings, and general technology firms lacking core GenAI offerings (e.g., Apple, Netflix). Chinese firms (e.g., Alibaba, Tencent, Baidu) are also excluded due to market access

limitations (16) and restricted financial transparency due to the Variable Interest Entity (VIE) structure of the BATX (Baidu, Alibaba, Tencent, and Xiaomi) stocks (4).

This scope ensures the system operates within a clearly bounded, analytically defensible domain. It enhances the relevance of recommendations by aligning them with user intent and facilitates reproducible evaluation by focusing on transparent and publicly accessible data.

# 2 Literature Review

The rapid advancement of Generative AI (GenAI) and its recent commercialisation in late 2022, has sparked significant interest in its application to financial forecasting and investment support systems. Despite notable progress in predictive modelling, current financial advisory tools often prioritise performance metrics at the expense of transparency and interpretability. This trade-off presents a particular challenge for non-technical retail investors, who require not only accurate but also comprehensible recommendations in order to make informed decisions.

This literature review examines recent developments at the intersection of GenAI, explainable AI (XAI), and algorithmic trading. It explores generative models for financial prediction, the integration of XAI techniques in forecasting systems, and the implications of user-centred design in FinTech. The aim is to critically evaluate existing approaches and identify a gap in the current landscape: the lack of accessible, domain-specific, and explainable investment tools. The review concludes by outlining how the *GenAI Advisor* project is designed to address this gap through a hybrid architecture that integrates transparent rule-based logic, supervised machine learning, and natural language explanation layers.

## 2.1 Generative AI in Financial Forecasting

The application of Generative AI (GenAI) in financial domains has emerged as a frontier area of research, particularly in the context of algorithmic trading and decision support. While traditional predictive systems rely on time-series models and supervised learning techniques, recent work has demonstrated the viability of large language models (LLMs) and generative transformers for tasks such as stock sentiment analysis, earnings prediction, and trade signal generation.

A notable example is *StockGPT* by Mai (9), which applies a fine-tuned generative language model to extract predictive signals from financial text and historical data. The system shows promising results in backtesting but offers limited transparency in terms of feature attribution and decision rationale. This highlights an emerging trade-off in GenAI systems between performance and interpretability, particularly in high-stakes domains like finance.

Generative models have also been used to simulate synthetic market data (15), enabling improved model training under data scarcity. However, these approaches often lack integration with rule-based or statistical baselines, making it difficult to assess the marginal utility of the generative component. Furthermore, the absence of human-centred explanation interfaces in these systems poses a barrier to adoption among non-specialist users.

These limitations underline the need for hybrid architectures that balance the predictive power of generative models with the interpretability of traditional techniques. The *GenAI Advisor* addresses this by separating signal generation from explanation, thereby ensuring that the core decision logic remains auditable while leveraging GenAI to enhance accessibility and user trust.

## 2.2 Explainable AI in Financial Systems

As AI-driven systems increasingly influence financial decision-making, the demand for explainable artificial intelligence (XAI) has become a critical concern. In domains characterised by uncertainty, regulatory oversight, and end-user scepticism, opaque "black-box" models are often ill-suited for practical deployment (6). This is particularly true for retail investors who may lack technical backgrounds yet require justifiable, transparent investment reasoning.

Marey et al. (10) propose an empirical framework that integrates deep learning with local explanation techniques, such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations). Their study demonstrates that user trust and engagement are positively correlated with the presence of intelligible model rationales, especially

when users are allowed to compare multiple explanation formats. However, these explanation mechanisms are typically appended to existing models rather than embedded in the design, leading to potential inconsistencies between what the model decides and what is communicated.

Other work, such as Ribeiro et al. (12), highlights the trade-off between global accuracy and local interpretability. While LIME-style approximations offer useful insight into model behaviour, they can misrepresent model logic under certain conditions, particularly in non-linear financial regimes.

To address these issues, the *GenAI Advisor* integrates XAI at both the structural and presentation layers. Rule-based logic provides a baseline of transparent, auditable recommendations, while machine learning classifiers offer enhanced signal detection. A local language model translates system outputs into natural-language explanations, preserving alignment between system behaviour and user-facing justifications. This layered approach ensures both decision accuracy and explanatory coherence.

### 2.3 Rule-based vs Machine Learning Investment Strategies

Investment recommendation systems have traditionally relied on rule-based logic grounded in technical indicators such as moving averages, momentum oscillators, and price-volume trends. These heuristics offer a high degree of interpretability, making them attractive to both practitioners and novice investors (11). Strategies based on conditions like moving average crossovers or Relative Strength Index (RSI) thresholds are easily understood and lend themselves to direct explanation.

However, rule-based systems often fail to capture the non-linearities and temporal dependencies inherent in financial time series. This has led to a growing interest in machine learning (ML) models for stock selection and signal generation, such as Random Forests, Gradient Boosting Machines (e.g. XGBoost), and neural networks (5). These models can discover subtle patterns and adapt to changing market regimes, offering superior predictive performance in some contexts.

Yet, the adoption of ML in retail investment contexts is hindered by concerns over transparency and robustness. Ryll and Seidens (13) highlight that black-box models frequently suffer from overfitting, poor generalisation, and limited interpretability when applied to financial data. Chakraborty and Joseph (3) similarly observe that financial institutions remain cautious about deploying ML in high-stakes environments due to its opacity and the risk of unpredictable behaviour under changing market conditions. These issues are especially problematic for non-expert users, who rely on trustworthy and intelligible decision support systems.

The *GenAI Advisor* bridges this gap by employing a hybrid strategy engine. Transparent rules serve as a reliable baseline, while supervised learning models (Random Forest and XGBoost) are used to enhance signal strength by capturing complex interactions. This dual approach enables comparative evaluation and supports user trust, as each recommendation is either traceable to a logical rule or supported by model-driven evidence.

### 2.4 Designing for Non-Technical Users in FinTech

The usability and accessibility of financial technologies remain a significant challenge, particularly for non-technical users who may lack domain expertise in data science or quantitative trading. FinTech interfaces that prioritise raw performance often neglect explainability and user-centred design, which are essential for effective decision support and adoption among retail investors. Ben David et al. (2) demonstrate through experimental evidence that the inclusion of explainable AI elements significantly improves trust and adoption rates among users of financial algorithmic advisors. Their findings underscore the critical role of intelligibility and transparency in interface design for enhancing user engagement.

Studies have shown that users without a financial background exhibit higher levels of anxiety and lower confidence when interacting with complex or opaque recommendation systems. Without clear explanations, users may disregard otherwise effective recommendations, especially in high-stakes settings involving personal investments. This issue is compounded by the fact that many retail-focused platforms do not disclose the reasoning behind their advice, instead presenting deterministic outcomes with minimal context (17).

Explainable user interfaces (XUIs) have been proposed as a solution, incorporating elements such as visual signal annotation, confidence metrics, and natural-language summaries (2). These approaches have demonstrated success in increasing perceived trust, interpretability, and willingness to act upon AI-generated recommendations. However, implementation of such systems is often inconsistent, and few are tailored to specific thematic portfolios like those in GenAI-focused sectors.

The *GenAI Advisor* responds to this gap by embedding a natural language explanation layer that interprets rule-based and ML-driven signals into plain-English rationales. By adopting a language-first, model-agnostic approach, the system enables users to engage meaningfully with its outputs, fostering transparency and informed decision-making without requiring technical fluency.

### 2.5 Backtesting and Evaluation in FinTech

Robust evaluation is a critical component of financial algorithm design, ensuring that investment strategies are not only theoretically sound but also empirically validated under realistic market conditions. Backtesting a strategy using historical data—remains the standard approach for evaluating financial models. However, its reliability depends on careful control for biases such as lookahead error, data snooping, and overfitting (1).

Evaluation metrics in the FinTech domain extend beyond accuracy to include financial risk/return measures. The Sharpe Ratio, for instance, assesses risk-adjusted return by penalising volatility, while maximum drawdown quantifies peak-to-trough losses, both essential for comparing the robustness of competing strategies (8). Other common metrics include total return, alpha and beta coefficients, and classification-based indicators such as precision, recall, and F1-score when using ML classifiers for signal prediction.

Tools such as `backtrader` and `pandas`-based frameworks facilitate integration of financial indicators, portfolio simulation, and metric reporting, making them essential in contemporary algorithmic finance development. Nevertheless, evaluation practices remain inconsistent, with many academic and commercial models reporting high in-sample performance without adequate out-of-sample or cross-validation testing (7).

The *GenAI Advisor* explicitly incorporates backtesting into its pipeline using reproducible Python-based workflows. It compares rule-based and machine learning strategies across multiple indicators and time frames, reporting both financial and statistical performance metrics. This dual-layered evaluation, provides transparency and supports empirical justification for investment recommendations.

### 2.6 Synthesis and Research Gap

The review of existing literature reveals a rich landscape of innovation at the intersection of AI, finance, and user-centred design. However, it also exposes clear limitations in current approaches when evaluated through the lens of accessibility, explainability, and domain specificity, three attributes that are critical for supporting non-technical investors seeking exposure to GenAI-focused equities.

Generative models such as *StockGPT* demonstrate that large language models (LLMs) can be fine-tuned to extract financial signals from unstructured data, yet their outputs often lack interpretability and alignment with human-understandable trading logic (9). While this presents

a promising avenue for performance gains, it simultaneously creates a transparency deficit. Similarly, Takahashi and Mizuno (15) propose using diffusion models to generate synthetic financial time series that replicate complex market behaviours. Although valuable for addressing data scarcity and irregularities, these techniques rarely incorporate explanatory mechanisms or support integration with domain-informed baselines, limiting their applicability in systems designed for human-in-the-loop decision making.

Research on XAI has provided a broad array of model-agnostic tools such as SHAP and LIME (12), which can help reveal internal model dynamics. Yet, as highlighted by Marey et al. (10), these methods are frequently bolted onto systems post hoc and suffer from a mismatch between model behaviour and user comprehension. Additionally, the form and delivery of explanations have not been standardised across FinTech applications, leaving a usability gap that affects trust and adoption.

This concern is amplified when considering ML-only investment systems. Despite their performance advantages, such models are often developed without sufficient consideration for interpretability or operational auditability. Ryll and Seidens (13) highlight that machine learning models in financial forecasting frequently suffer from overfitting and poor generalisation, particularly under changing market regimes. Similarly, Chakraborty and Joseph (3) caution that opaque model architectures pose a challenge for real-world deployment in financial institutions, where accountability and stability are essential. Conversely, rule-based systems offer clarity and transparency but often lack the adaptability required to navigate complex, non-linear market dynamics.

From a usability perspective, research indicates that FinTech platforms frequently underestimate the cognitive demands placed on users, often failing to present decision rationales in a digestible and actionable format. Ben David et al. (2) demonstrate that the inclusion of explainable elements—such as natural-language justifications and transparency cues, significantly improves user trust and the likelihood of adopting AI-generated financial advice. While some interfaces include features like confidence bands or risk scores, few extend to personalised, intelligible explanations grounded in model logic and user context. Moreover, no existing platform explicitly addresses the needs of investors seeking targeted exposure to the GenAI sector, a growing interest area underserved by conventional robo-advisors.

In terms of evaluation, there is a consistent pattern of insufficient out-of-sample validation and limited comparative benchmarking across strategy types (1; 7). Financial metrics such as Sharpe Ratio or drawdown are inconsistently applied, and many studies prioritise classification accuracy without accounting for investor risk tolerance or decision thresholds.

Taken together, these findings point to a clear research and development gap: there is currently no publicly available tool that offers GenAI-specific stock recommendations through a hybrid system combining auditable rules, machine learning models, and explainable outputs tailored to non-expert users. Moreover, there is a lack of reproducible, modular pipelines that support comparative evaluation of rule-based and ML-based strategies in the context of GenAI sector investment.

The *GenAI Advisor* aims to fill this gap by providing an integrated, open-source prototype that embodies the insights drawn from this literature. It does so through: (1) a hybrid strategy engine that combines clarity and complexity; (2) a local LLM explanation layer to generate user-friendly rationales; (3) focused stock screening based on GenAI sector relevance; and (4) a rigorous evaluation framework using both financial and statistical metrics. By unifying these components, the system advances the current state of practice in explainable FinTech design for thematic investing.

# 3 System Design

This chapter presents a detailed account of the design of the GenAI Advisor system, designed to deliver explainable, user-friendly financial insights for retail investors with an interest in Generative AI equities.

## 3.1 System Overview & Design Rationale

The architecture of the GenAI Advisor system is composed of five distinct yet interdependent layers:

1. **Data Pipeline**, responsible for sourcing, cleaning, and transforming financial data into formats suitable for analysis.
2. **Strategy Engine**, the analytical core that integrates retrieved knowledge to generate structured investment insights.
3. **Explanation Generator**, producing justifications and rationales for each recommendation.
4. **User Interface**, delivers an intuitive interaction point for retail users, visualising results in a digestible and accessible format.
5. **Backtesting & Evaluation**, supports validation of generated recommendations through historical simulations, user feedback loops and performance scoring.

Each layer encapsulates a single responsibility while remaining interoperable with adjacent layers, reflecting a separation-of-concerns design philosophy. This modular approach enables independent development and testing of each component, reducing coupling and simplifying future iterations or extensions.

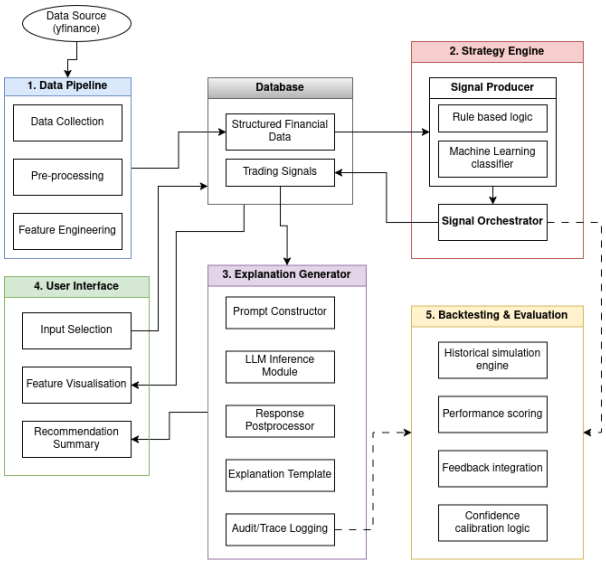


Figure 2: Five-layer Architecture of GenAI Advisor

The design rationale also reflects broader system requirements, particularly the need for explainability, adaptability, and integration with real-time financial data streams. Embedding traceability into the architecture ensures that every recommendation is grounded in verifiable data sources, supporting responsible AI usage. Furthermore, by situating the explanation generator as a distinct layer, the architecture highlights the importance of user trust and transparency as core design principles.

Scalability is not a goal of the current system, and no scaling mechanisms will be implemented in either the prototype or the final version. However, scalability remains an important consideration in the broader context of deploying AI systems in production environments. Should the system be developed further, future work could explore stateless service design, containerised deployment, and asynchronous processing to support larger data volumes and concurrent users. Flexibility across infrastructure backends and compatibility with different classes of language models could also be considered to enable wider applicability.

### 3.2 Data Pipeline

This foundational layer is responsible for acquiring, preparing, and transforming financial data into structured formats that support both retrieval and modelling. It ensures that the inputs to the system are consistent and semantically rich.

**Data Sources:** Financial data is sourced via the `yfinance` Python library, which extracts structured data from Yahoo Finance. This includes:

- Daily price data
- Adjusted close prices accounting for dividends and splits
- Market fundamentals including P/E ratio, EPS, market capitalisation, and beta
- Dividend history, earnings results, and analyst forecasts
- Corporate actions such as stock splits and ex-dividend dates

**Pre-processing:** All raw data is immediately pre-processed in memory using `pandas` and associated libraries before being persisted. This includes:

- Handling of missing values and filtering of anomalous entries
- Normalisation and transformation of numerical columns
- Resampling and forward-filling to align time indices across features
- Annotating rows with relevant metadata

This pre-processing ensures consistency across all instruments and time periods and reduces computational load in downstream stages.

**Data Storage:** Once cleaned and transformed, the processed dataset is stored in a relational MariaDB database. This allows for fast, structured access by subsequent system components and avoids repeated API calls to Yahoo Finance. Tables are designed to support efficient filtering by date, ticker, and feature type. Indexes are maintained on primary keys and time columns to optimise query latency.

**Feature Engineering:** This stage derives higher-order signals and representations from the cleaned data:

- **Technical indicators:** including moving averages, MACD, RSI, and volatility bands
- **Fundamental deltas:** e.g. growth rates in earnings, changes in valuation multiples
- **Risk signals:** calculated from rolling variance, beta, and return dispersion
- **Embeddings:** tabular records are converted into dense vector representations using Sentence-BERT for semantic similarity and retrieval

**Design Justification:** The choice of MariaDB for structured data storage reflects a balance between transparency, performance, and system complexity. Relational databases offer well-defined schema enforcement, support for indexed queries, and seamless integration with Python data analysis libraries. This aligns with the project's emphasis on explainability, traceability, and reproducibility. Alternative solutions were considered, including DuckDB for embedded analytics or Parquet-backed storage for columnar efficiency. However, these options introduce either added complexity or are optimised for larger-scale, concurrent systems. For the scope of this project, focusing on iterative prototyping, backtesting, and clarity, MariaDB provides a robust and pedagogically sound foundation.

### 3.3 Strategy Engine

The strategy engine forms the analytical core of the GenAI Advisor system, responsible for evaluating investment opportunities using structured signals and domain-specific rules. It interprets features derived from historical data and generates intermediate outputs such as buy/sell/hold flags, risk scores, and confidence levels. This layer operates independently of any generative model.

**Signal Retrieval:** At runtime, the strategy engine queries the MariaDB backend for time-aligned features such as technical indicators, valuation metrics, volatility profiles, and momentum scores. Data is filtered by ticker, date, and sector using predefined criteria. Missing values are handled via forward-filling or exclusion depending on the indicator's sensitivity.

**Rule-based Scoring:** Initial decision logic is implemented using deterministic rules. For example:

- *Momentum signal:* Generate a buy signal if the closing price exceeds the 50-day simple moving average and RSI is in the range 50–70.
- *Valuation screen:* Flag as overvalued if the P/E ratio is greater than the sector median by more than one standard deviation.
- *Risk classification:* Assign a "high risk" label if beta exceeds 1.5 or if recent earnings variance is above a defined threshold.

These rule sets are configurable and grounded in established financial heuristics.

**Machine Learning Models:** Where applicable, supervised models such as logistic regression or gradient-boosted decision trees may be used to estimate directional signals. Input features include moving averages, volatility, return profiles, and sector-adjusted valuation ratios. Outputs are binary or probabilistic classifications of investment suitability.

**Strategy Output:** The final output is a structured intermediate recommendation, for example:

- *Ticker:* MSFT
- *Action:* HOLD
- *Confidence:* 72%
- *Tags:* "Moderate Risk", "Overvalued"

This output is passed to the explanation layer, where it is contextualised and communicated to the end-user through a natural language interface.

The strict separation of decision logic from explanation ensures that recommendations are reproducible and auditable across time.

### 3.4 Explanation Generator

The explanation generator is responsible for converting structured strategy outputs into human-readable narratives. This layer interfaces with a large language model (LLM) to articulate the rationale behind each recommendation, ensuring the system remains explainable, and transparent.

**Prompt Construction:** Each recommendation generated by the strategy engine is converted into a structured prompt. This prompt contains:

- The recommended action (e.g. BUY, HOLD, SELL)
- Associated features (e.g. P/E ratio, RSI, volatility)
- Descriptive tags (e.g. "Overvalued", "Moderate Risk")
- Ticker metadata (e.g. sector, market cap, recent price movement)

The prompt template enforces a consistent narrative structure and ensures that only relevant, pre-computed data is surfaced to the LLM. No raw retrieval or document inference is performed at this stage.

**LLM Invocation:** The prompt is submitted to a locally hosted language model, with inference restricted to models that can be executed on a GPU with 8GB of VRAM. This design constraint promotes offline operability.

**Post-processing and Attribution:** The LLM output is parsed and linked to its underlying features. Numerical drivers are tagged and cross-referenced with the original strategy inputs to enable traceability. Where applicable, confidence indicators and relevant feature values are surfaced alongside the textual response.

**Model Selection Rationale:** For the purpose of this project, fine-tuning large language models is deliberately excluded due to its high resource demands and limited added value in the context of structured explanation generation. Instead, the system leverages pre-trained models served locally via the Ollama framework, which supports quantised variants of open-source architectures such as Mistral, Phi-2, and LLaMA2. These models strike a practical balance between performance and hardware feasibility, running comfortably on an 8GB VRAM GPU. The focus is placed on prompt design and response evaluation rather than model adaptation. This strategy aligns with the project's emphasis on transparency and reproducibility, while avoiding the complexity and risks associated with custom model training.

By isolating the explanation function from decision-making logic and constraining it to local, auditable models, this design ensures the advisory system remains intelligible, trustworthy, and compliant with practical deployment constraints.

### 3.5 User Interface

The user interface (UI) serves as the primary point of interaction between the end-user and the advisory system. It is designed to surface system outputs in a clear and context-aware format while abstracting away the technical complexity of the underlying architecture.

**Interface Design Principles:** The UI is guided by the principles of usability, transparency, and minimalism. It aims to provide information density without cognitive overload, and supports both exploratory queries and structured evaluations.

**Interaction Model:** The primary mode of interaction is a ticker selector and a time window. Queries are routed through the backend pipeline and responses are returned in astructured, explanation-rich format.

**Presentation Layer:** Output is displayed in modular response cards, with each card corresponding to a company. Each card includes:

- The recommended action (BUY, HOLD, SELL)

- A summarised rationale generated by the explanation layer
- Key features (e.g. P/E ratio, RSI, beta) highlighted inline
- Option to expand for full explanation and numerical context

Cards are sorted based on confidence or relevance, and users can collapse or pin cards for comparison.

**Technical Implementation:** The frontend is built using a lightweight web framework such as Streamlit. This ensures rapid prototyping, cross-platform compatibility, and support for secure API calls to backend services. The UI is deployed as a local web application accessible via browser.

**Responsiveness and Accessibility:** The interface is optimised for both desktop and mobile devices. Layouts use responsive containers and typography is chosen for readability. Accessibility guidelines such as contrast ratios and semantic labelling are followed to ensure inclusivity.

This interface completes the user-facing layer of the system, enabling seamless communication of insights derived from structured analysis and local language model generation.

### 3.6 Backtesting & Evaluation

This layer is responsible for assessing the performance, consistency, and interpretability of the GenAI Advisor system. Evaluation is conducted along two axes: (i) financial effectiveness of the recommendations and (ii) fidelity and stability of the generated explanations.

**Backtesting of Strategy Output:** The system supports retrospective evaluation of strategy outputs using historical financial data stored in MariaDB. Given a time window and a set of tickers, the engine replays strategy decisions based on data available at the time. The simulated trades are benchmarked using standard performance metrics such as:

- Cumulative return
- Sharpe ratio
- Maximum drawdown
- Hit rate (proportion of correct directional calls)

These simulations validate whether the rule-based and ML-driven components produce economically viable signals under realistic conditions.

**Evaluation of Explanation Consistency:** Because the system includes a generative component, it is essential to assess the consistency and fidelity of explanations. For a given strategy output, repeated LLM invocations are evaluated for semantic agreement and factual stability. Techniques include:

- *Stability under identical prompts:* The same input is submitted multiple times to check for drift.
- *Paraphrase testing:* Slightly altered prompts are compared for explanation robustness.
- *Key fact alignment:* Extracted metrics in the explanation are compared against known strategy outputs.

These tests ensure that the narrative remains faithful to the underlying numerical reasoning.

**User Feedback Loop:** The UI logs user reactions to recommendations through binary feedback (e.g. thumbs up/down). Feedback is stored with associated prompt and strategy metadata, enabling post-hoc review and potential retraining of classification thresholds or strategy weights.

**Qualitative Auditing:** In addition to automated tests, a sample of responses is manually audited to assess:

- Clarity of the natural language explanation
- Justification coverage (i.e., do the reasons align with the action?)
- Absence of hallucinated content or misleading terminology

This step supports ethical AI principles by ensuring outputs remain comprehensible and appropriate for retail consumption.

This evaluation layer closes the loop between system logic, narrative explanation, and end-user experience. It provides the foundations for continuous improvement while maintaining interpretability and trust.

# 4 Implementation

## 4.1 Overview

The GenAI Advisor was developed using a Minimum Viable Product (MVP) approach, structured around four modular components: data ingestion, strategy engine, explanation generation, and a Streamlit-based user interface. The system was designed to be fully offline, prioritising data privacy, transparency, and reproducibility. Initial efforts focused on building a thin end-to-end pipeline to demonstrate feasibility and guide future sprints.

## 4.2 System Architecture

The overall system architecture comprises:

- **Frontend:** Streamlit application for user interaction
- **Backend:** Modular Python packages handling data ingestion, strategy evaluation, and explanation generation
- **LLM Inference:** Mistral 8B model run locally via Ollama
- **Testing and Evaluation:** Pytest-driven unit tests and backtesting utilities

A modular layout allowed for parallel development and easier testing of individual system components.

## 4.3 Development Process

An MVP-first approach guided the early stages. Agile-inspired sprints were used to iteratively add functionality and refine components. Git was used for version control, and all new modules were tested before integration. (https://github.com/andreasmallios/genaiadvisor)

## 4.4 Key Modules and Features

### 4.4.1 Data Ingestion

Equity price data is retrieved from Yahoo Finance via the `yfinance` API, using a custom function designed to streamline data access and ensure reproducibility. The function `fetch_ticker_data` encapsulates logic for both online retrieval and local caching. When data for a given ticker symbol already exists in the local cache (CSV format), it is loaded directly using `pandas.read_csv`, with the `Date` column parsed into a datetime index to facilitate time series analysis. This not only minimises reliance on external API calls, but also enhances the determinism of backtests.

In the absence of cached data, the function instantiates a `yfinance.Ticker` object and invokes `Ticker.history()` with a default window of one year and daily frequency. Metadata fields unrelated to price analysis (e.g., `Dividends`, `Stock Splits`) are explicitly excluded to avoid downstream noise. The retrieved data frame is then persisted locally to disk, ensuring future runs use identical input data unless manually refreshed.

All data is indexed by date, with explicit naming of the index to `Date` for consistency across modules. Additionally, the local directory used for caching is dynamically resolved using Python's `os.path` utilities, enabling portability across environments. Although not shown directly in the function, timezone consistency is enforced at the point of usage through `tz_localize`, ensuring temporal coherence across the ingestion and backtesting pipeline.

### 4.4.2 Strategy Engine

The strategy engine was originally implemented using a simple moving average (SMA) crossover, a well-known momentum indicator. However, it was later modularised and extended into a hybrid ensemble system, integrating both traditional technical indicators and machine learning predictions. Each strategy component was abstracted into its own module with a shared interface returning a `recommendation`, `reason`, and `date`, thereby enabling flexible composition and detailed introspection.

The indicators implemented include SMA crossover, Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger Bands, and the Stochastic Oscillator. Each module performs well-defined signal computation:

- **SMA crossover**: Computes short- and long-term SMAs (50-day and 200-day by default) and identifies bullish or bearish crossovers as trading signals.
- **RSI**: Quantifies momentum and detects overbought or oversold regimes, issuing BUY signals when RSI falls below 30 and SELL when above 70.
- **MACD**: Uses exponential moving averages to detect momentum shifts via signal-line crossovers.
- **Bollinger Bands**: Employs a 20-day SMA and standard deviation bands to detect volatility-based breakouts.
- **Stochastic Oscillator**: Evaluates recent closing prices relative to the high-low range over a rolling window to spot potential reversals.

Additionally, a supervised machine learning layer was incorporated using both Random Forest and Logistic Regression classifiers. These were trained on historical technical features, including SMA differentials, RSI, MACD, returns, and volume, with future returns used as labels. Predictions from both models are aggregated and translated into BUY, HOLD, or SELL signals.

All individual signals are passed to a meta-layer defined in `engine.py`, where a weighted voting mechanism combines the recommendations. Each strategy is assigned a normalised weight, and the aggregate score is computed to produce the final advisory signal. A confidence score is also generated, which allows the user to assess the strength of the consensus recommendation.

This modular and explainable architecture enhances the system's adaptability, allowing future indicators or models to be integrated with minimal changes to the existing pipeline.

### 4.4.3 Explanation Generator

To enhance transparency and user trust, the system includes a natural language explanation generator powered by a locally hosted instance of the Mistral 8B model, served via Ollama. This design deliberately avoids reliance on proprietary cloud APIs, enabling offline inference and preserving data privacy. Communication with the model is facilitated through Python's `subprocess` module, which programmatically invokes the model via command-line interface.

Prompts are dynamically constructed using the metadata associated with each final recommendation. The resulting prompt embeds a structured format with three components: *Summary*, *Reason*, and *Action*. These guide the model to produce consistent, human-readable rationales tailored for non-specialist audiences. The prompt instructs the model to contextualise signals such as SMA crossover, RSI, MACD, Bollinger Bands, Stochastic Oscillator, and ML Classifier outputs. It also emphasises educational tone, British English, and a strict 300-word limit to maintain clarity and concision.

By embedding these constraints in the prompt and invoking the model locally, the system achieves explainability without introducing dependencies that compromise reproducibility

or user autonomy. Errors in generation are handled gracefully, returning informative fallback messages to aid debugging.

#### 4.4.4 Interface

The user interface is developed using Streamlit, a Python-based rapid application framework tailored for data-centric interfaces. This decision facilitated fast prototyping and interactive visualisation, crucial during iterative development and debugging. The application enables users to input stock tickers, retrieve historical market data, view system-generated BUY, HOLD, or SELL recommendations, and inspect the rationale behind each decision.

The main interface consists of two parts: a portfolio overview and a detailed ticker analysis. The portfolio table displays summarised recommendations for all tickers within the watchlist, fetched from a CSV file. For deeper inspection, users may select a specific ticker, upon which the interface renders price history, a modular signal breakdown, and a natural language explanation. This layered interaction supports transparency and interpretability.

Behind the scenes, the app integrates tightly with the data ingestion pipeline, strategy engine, and explanation generator. Recommendations and signal details are fetched dynamically and rendered as structured text. Visual aids, such as historical closing price charts, are plotted using `matplotlib` and embedded into both the interface and downloadable PDF reports. This PDF generation, facilitated by the `fpdf` library, enables offline review and archiving.

Importantly, the interface includes clear disclaimers and educational prompts, ensuring compliance with ethical guidance and reinforcing the tool's non-advisory role. Collectively, the Streamlit-based interface serves not only as a functional front-end, but also as a pedagogical instrument that demystifies algorithmic financial decision-making for non-experts.

### 4.5 Implementation Challenges

Several technical challenges emerged during implementation, each requiring targeted debugging and design intervention.

- **Timezone mismatch:**
  Historical price data retrieved via Yahoo Finance lacked explicit timezone metadata, which introduced ambiguity when slicing datasets for backtesting. This was particularly problematic when aligning cut-off dates across modules. To mitigate this, all datetime indices were explicitly localised to the `America/New_York` timezone using the `tz_localize` method. This normalisation ensured consistent interpretation of trading days and eliminated subtle off-by-one errors during data slicing.

- **CSV integrity:** Cached datasets were initially stored as CSV files to improve reproducibility and reduce API reliance. However, inconsistencies in how indices were handled during `to_csv()` and `read_csv()` operations led to misaligned time series and malformed DataFrames. This was addressed by explicitly setting the `index_label` parameter during write operations and enforcing `index_col="Date"` with `parse_dates=True` during reads. These changes ensured referential consistency across modules and safeguarded downstream processing.

- **Subprocess inference:** Local model inference via the `ollama` CLI introduced challenges in handling prompt formatting, encoding, and subprocess errors. The prompt had to be programmatically composed with structured constraints while maintaining compatibility with standard input streams. Errors during model invocation (e.g., non-zero return codes) were caught and reported via Python's `subprocess.CalledProcessError` exception handling, allowing the interface to degrade gracefully with informative fallback messages.

- **Test discovery:** During unit test integration, Pytest failed to resolve relative imports due to the project's nested directory structure. This was resolved by invoking Pytest with the environment variable `PYTHONPATH=.`, thereby ensuring that all modules were discoverable relative to the project root. This solution avoided the need for path rewrites or package restructuring.
- **Backtesting leakage:** Initial backtest prototypes inadvertently included post-cutoff data in model inputs, leading to optimistic and invalid evaluations. This was corrected by enforcing a strict temporal cutoff: recommendations were generated using only data up to the target date, while subsequent price movements were isolated for outcome analysis. This approach ensured methodological rigour and prevented data leakage, a critical consideration in empirical evaluation.

These challenges highlight the complexity of integrating financial data pipelines, local LLM inference, and modular testing in a cohesive system. Their resolution contributed directly to the robustness and reproducibility of the final implementation.

# 5 Evaluation

Following the successful construction of the initial MVP, a systematic evaluation was carried out to assess the correctness, reliability, and effectiveness of the GenAI Advisor as advanced features were layered onto the system.

The first step in this evaluation phase was the implementation of a modular strategy engine architecture, enabling the integration of additional signals including RSI (Relative Strength Index) and MACD (Moving Average Convergence Divergence), alongside the existing SMA crossover strategy. Each of these signals was designed as a modular Python component, facilitating independent unit testing and allowing for clear traceability of each signal's contribution to the overall recommendation engine.

Test-driven development practices were applied rigorously using `Pytest`, ensuring that the advanced signals operated correctly and that the recommendation engine's logic for combining multiple signals into final recommendations remained consistent and transparent. Tests confirmed that the system produced valid outputs under a variety of market conditions, providing confidence in the robustness of the extended MVP.

To evaluate the practical performance of the recommendation signals, a batch backtesting pipeline was developed, allowing the system to be tested systematically across multiple tickers and historical dates with a defined lookahead period. This pipeline produced structured CSV outputs containing recommendations and subsequent price movements, providing a clear dataset for analysis.

An evaluation notebook was prepared using Jupyter, processing these CSV outputs to visualise the distribution of price movements following the system's recommendations and to analyse the effectiveness of each signal type in isolation and in combination. Plots such as histograms and boxplots were generated to illustrate the distribution of returns following recommendations, and initial quantitative metrics were computed, such as the proportion of BUY signals that resulted in positive returns.

These evaluation steps ensured that the extended MVP did not introduce regressions while adding advanced functionality and that it continued to align with the goals of interpretability, offline operation, and explainability. The evaluation process also provided data-driven evidence to support further refinement of signal thresholds, signal weighting, and prompt engineering for explanation generation in subsequent development sprints.

### 5.0.1 Testing and Evaluation Support

To ensure the correctness and robustness of the implementation, the system was instrumented with unit tests using the `pytest` framework. Each functional module—including data ingestion, technical indicator computation, and explanation generation—was covered by dedicated tests that validated outputs against structural and semantic expectations. For instance, the SMA, RSI, MACD, Bollinger Bands, and Stochastic Oscillator modules were tested for valid signal outputs and consistent return types. The explanation generator was similarly validated to ensure that LLM-generated text adhered to format and length constraints.

Beyond unit testing, an evaluation pipeline was implemented via a custom backtesting framework. Historical ticker data was sliced as of specified past dates, and the system was queried to generate time-local recommendations and explanations. These were then evaluated against forward price movements over configurable horizons (e.g., 30 days), enabling ex-post performance analysis. A batch wrapper automated this process across multiple tickers and time windows, with results persisted in timestamped CSVs for downstream analysis and visualisation in Jupyter notebooks.

This combined approach—functional testing and historical simulation—ensured both implementation integrity and empirical grounding. By decoupling test targets and maintaining test coverage alongside development, the system achieved high maintainability and evaluation

traceability.

# 6 Conslusion

lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## 6.1 Summary and Forward Plan

The implemented system constitutes a functional Minimum Viable Product (MVP) that integrates data ingestion, technical signal computation, ensemble logic, and explainable recommendation generation through a cohesive interface. This modular architecture facilitated rapid prototyping, iterative evaluation, and extensibility across system components. The ability to trace recommendations back to interpretable signals and test outputs in a reproducible manner established a strong foundation for further development.

Future work is centred around improving the system's expressiveness, adaptivity, and user experience:

- **Explanation pipeline enhancements:** Current prompts to the local language model are stateless. Incorporating context-aware memory—such as dialogue history, prior user queries, or preference metadata—would allow for more personalised and coherent multi-turn explanations. This may involve integrating a lightweight memory layer or using conversation-aware prompt templates to simulate state retention.

- **Empirical signal weighting:** The existing strategy engine applies fixed weights to each indicator. However, not all signals contribute equally to predictive accuracy across assets or timeframes. Future iterations will incorporate a data-driven signal weighting mechanism, potentially via logistic regression or Bayesian optimisation, based on historical backtest performance. This would enable adaptive ensemble construction and improved recommendation fidelity.

- **User interface improvements:** While the Streamlit interface effectively demonstrates system functionality, future UI enhancements will focus on interactivity and visual storytelling. This includes time-series overlays for each signal, interactive sliders to simulate portfolio scenarios, and user-driven feedback loops. Richer visualisations will also support educational transparency, a core design objective of the tool.

Collectively, these planned developments aim to deepen the system's interpretability, robustness, and real-world decision support potential—particularly for novice retail investors seeking algorithmic guidance that is both transparent and context-sensitive.

## References


[1] David H. Bailey, Jonathan M. Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu. 2014. The Probability of Backtest Overfitting. *Journal of Computational Finance* 20, 4 (2014), 39–69.

[2] Daphna Ben David, Yedidya S. Resheff, and Tal Tron. 2021. Explainable AI and Adoption of Financial Algorithmic Advisors: an Experimental Study. *arXiv preprint arXiv:2101.02555* (2021). https://arxiv.org/abs/2101.02555

[3] Chiranjit Chakraborty and Alex Joseph. 2017. *Machine Learning at Central Banks.* Staff Working Paper 674. Bank of England. https://www.bankofengland.co.uk/-/media/boe/files/working-paper/2017/machine-learning-at-central-banks.pdf

[4] Fa Chen. 2021. Variable interest entity structures in China: are legal uncertainties and risks to foreign investors part of China's regulatory policy? *Asia Pacific Law Review* 29, 1 (2021), 1–24. https://doi.org/10.1080/10192557.2021.1995229

[5] Thomas Fischer and Christopher Krauss. 2018. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research* 270, 2 (2018), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

[6] David Gunning and David Aha. 2019. DARPA's Explainable Artificial Intelligence (XAI) Program. *AI Magazine* 40, 2 (2019), 44–58. https://doi.org/10.1609/aimag.v40i2.2850

[7] I. Kaastra and M. Boyd. 1996. Designing a Neural Network for Forecasting Financial and Economic Time Series. *Neurocomputing* 10, 3 (1996), 215–236. https://doi.org/10.1016/0925-2312(95)00039-9

[8] Andrew W. Lo. 2002. The Statistics of Sharpe Ratios. *Financial Analysts Journal* 58, 4 (2002), 36–52. https://doi.org/10.2469/faj.v58.n4.2453

[9] D. Mai. 2024. StockGPT: A GenAI Model for Stock Prediction and Trading. *SSRN* (2024). https://ssrn.com/abstract=4787199

[10] N. Marey, A. A. Abu-Musa, and M. Ganna. 2024. Integrating Deep Learning and Explainable Artificial Intelligence Techniques for Stock Price Predictions. *International Journal of Accounting and Management Sciences* 3, 4 (2024), 479–504.

[11] John J. Murphy. 1999. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications.* New York Institute of Finance.

[12] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You? Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1135–1144.

[13] Lukas Ryll and Sebastian Seidens. 2019. Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey. *arXiv preprint arXiv:1906.07786* (2019). https://arxiv.org/abs/1906.07786

[14] S&P Global Market Intelligence. 2025. *GenAI funding hits record in 2024, boosted by infrastructure interest.* https://www.spglobal.com/market-intelligence/en/news-insights/articles/2025/1/genai-funding-hits-record-in-2024-boosted-by-infrastructure-interest-87132257 Accessed May 2025.

[15] Tomonori Takahashi and Takayuki Mizuno. 2024. Generation of synthetic financial time series by diffusion models. *arXiv preprint arXiv:2410.18897* (2024).

[16] U.S. Department of Commerce. 2022. *Commerce Implements New Export Controls on Advanced Computing and Semiconductor Manufacturing Items to the People's Republic of China (PRC).* Technical Report 2022-21658. Bureau of Industry and Security. `https://www.govinfo.gov/content/pkg/FR-2022-10-13/pdf/2022-21658.pdf` Federal Register, Vol. 87, No. 197, pp. 62186–62215.

[17] Alex Zarifis and Xusen Cheng. 2024. How to build trust in answers given by Generative AI for specific, and vague, financial questions. *arXiv preprint arXiv:2408.14593* (2024). `https://arxiv.org/abs/2408.14593`